LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-664250

Fully-Implicit Orthogonal Reconstructed Discontinuous Petrov-Galerkin Method for Multiphysics Problems

R. Nourgaliev, H. Luo, S. Schofield, T. Dunn, A. Anderson, B. Weston, J. Delplanque

November 13, 2014

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL-TR-664250

Fully-Implicit Orthogonal Reconstructed Discontinuous Petrov-Galerkin Method for Multiphysics Problems

R.Nourgaliev, H.Luo, S.Schofield, T.Dunn, A.Anderson,
B.Weston, and J.-P. Delplanque

February 18, 2015

This Page is Intentionally Left Blank

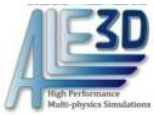
Fully-Implicit Orthogonal Reconstructed Discontinuous Petrov-Galerkin Method for Multiphysics Problems

ROBERT NOURGALIEV*, HONG LUO†, SAM SCHOFIELD*, TIM DUNN**,
ANDY ANDERSON**, BRIAN WESTON‡, JEAN-PIERRE DELPLANQUE‡

***Weapons & Complex Integration, B-Division**
****Computational Engineering Division**
Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, CA 94551, USA

†North Carolina State University,
Mechanical & Aerospace Engineering
Engineering Bldg 3, Campus Box 7910, 911 Oval Drive
Raleigh, NC 27695-7910, USA

‡Department of Mechanical and Aerospace Engineering,
University of California Davis,
Davis, CA 95616



LLNL-TR-664250 (202 p.)
February 18, 2015

This Page is Intentionally Left Blank

Abstract

THIS document is focussed on the development of reconstructed discontinuous Galerkin ($\mathrm{RDG}_{\mathrm{P_nP_m}}$) method, based on orthogonal basis functions. Orthogonality of basis functions is essential for enabling robust and efficient fully-implicit Newton-Krylov based time discretization. The method is designed for generic partial differential equations, including transient hyperbolic, parabolic and/or elliptic operators, which are attributed to many multiphysics problems. Our driving application of interest is additive manufacturing, which requires simulations of fluid-solid-gas systems, with compressibility effects and phase change (melting/solidification). Thus, we demonstrate the method capabilities for solving nonlinear heat conduction equation coupled with multi-material Navier-Stokes based compressible fluid (in low Mach number limit) systems. We focus our attention to the method accuracy (in both time and space), as well as solvability of linear algebra involved in linear steps of the Newton-based solver. We demonstrate the accuracy and efficacy of the method, emphasizing the advantages from the orthogonality of basis functions, which makes better conditioning of underlying (approximate) Jacobian matrices, and rapid convergence of Krylov methods.

This Page is Intentionally Left Blank

Acknowledgements

THIS work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, and funded by the Laboratory Directed Research and Development Program at LLNL under project tracking code 13-SI-002.

This Page is Intentionally Left Blank

Contents

This Page is Intentionally Left Blank

List of Tables

This Page is Intentionally Left Blank

List of Figures

This Page is Intentionally Left Blank

Chapter 1

Introduction

In this document, we summarize the development of fully-implicit solution algorithm, based on the Reconstructed Discontinuous Galerkin (RDG) method. The main technical contribution is the use of orthogonal basis functions, which is aimed at better conditioning of linear steps during the Newton-based non-linear iterative procedure. In addition, we demonstrated an ability to solve for primitive variables, chosen on the requirement of solvability (better conditioning) of the underlying physics. This is in contrast to our previous RDG efforts ([LBL08], [LBL06], [LLNM10b], [LLNC11], [LLN12], [LXL+12], [LXS+13], [XLFN14], [XLN14] [LLNM09], [LLNM10c]), which was using conservation variables as a solution vector. In the context of application of interest, we are looking at **all-speed flow** capabilities, with **phase change (melting/solidification)**, and the sets of primitive variables ensuring solvability involve pressure, velocity and specific internal energy/enthalpy/temperature. In this cases, the set of conservation variables (i.e., mass, momentum and total energy) is poorly conditioned and restrictive in terms of feasible flow regimes (Mach number limitations).

The work is motivated by applications in Additive Manufacturing (AM), requiring resolution of the laser-induced powder melting and formation of liquid metal pools, with numerous numerically challenging issues. These include modeling of liquid-solid interfaces, due to powder melting/solidification; gas-solid and gas-liquid multi-material interfaces, with representation of ambient gaseous media (air, Argon), using fully-compressible formulation; and complex interfacial physics, including surface tension and Marangoni effects, adequate representation of fluid-solid-gas contacts and wetting phenomena, etc. In this study, we will be focusing on all-speed flow capabilities and an adequate modeling of phase change.

To enable numerical resolution of all speed flow regimes, and stiffness associated with material strength models, covering both liquid (Newtonian fluid viscous stress tensors) and solids (with appropriate stress deviators, plastic strain and other constituitive models), we will be working with fully-implicit solvers, which are based on tight coupling of all involved physics. While we are interested in resolving slow dynamic time scales of the process (mostly at the material velocity time scale), fast (stiff) time scales due to acoustics and material strength are also properly accounted for.

We will capitalize on recent development in L-stable time integrators [BCVK02, CKB+05] and Newton based multiphysics algorithms [KK04], which is in contrast to Picard iteration based fully-implicit solution algorithms[1] [PS72, Pat80], [Iss85, OI01], currently widely used in commercial fluid dynamics codes, such as CFX, STAR-CCM+, Fluent, etc., and many research and open-source codes, such as e.g. OpenFoam.

For space discretization, we will use recently developed Reconstructed Discontinuous Galerkin method, which is shown to be highly attractive approach for high-order numerical discretization of multi-physics problems ([LBL08], [LBL06], [LLNM10b], [LLNC11], [LLN12], [LXL+12], [LXS+13], [XLFN14], [XLN14] [LLNM09], [LLNM10c]). Our emphasis here is placed on making the RDG to robustly work within the fully-implicit framework, especially at the regimes when the underlying linear algebra is highly stiff. For this purpose, we choose to operate with orthogonal basis functions, which supports better conditioning of time discretizators. To ensure orthogonality, we use modal DG with Legendre based tensor-product basis functions. In addition, we modify test functions, by inverse-Jacobian weighting the basis functions, which places the method within the general class of Petrov-Galerkin formulations. Thus constructed RDG has numerous attractive features. Beside orthogonality, the method is hierarchical, which enables natural compatibility with p-refinement. The method inherits all advan-

[1] It is instructive to note that we are excluding from the consideration *operator-splitting methods*, such as *projection methods* [Cho68], [Kan86], [BCG89], [Gre90, GC90, GCCH95, GC96], [ABCH93, ABS96, ABC00], [Rid94b, Rid94a, RKM+95, Rid95], [Min96], [GQ97], [PAB+97], [SAB+99], [KNW99], [BM95, MB97], [Wet98], [Gue96, Gue97], [GQ98b, GQ98a], [ABC00]; and *"Implicit Continuous-fluid Eulerian (ICE)"* algorithm, [HA68, HA71, HA75b, HA75a]). These methods are very successful in simulation of relatively simple fluid dynamics, but are considered not robust enough for application of interest in the present study.

tages from the original RDG, including easy implementation on hybrid meshes and an ability to work naturally with AMR. Since the base (zeroth-order) degrees of freedom are cell-average quantities, the method should be viewed as DG-based extension of the second-order finite-volume (FV) algorithm, to enable high-order ($> 2^{\text{nd}}$) discretization on unstructured hybrid meshes, without extension of the stencil. In fact, the stencil of the RDG is exactly the same as in the second-order finite-volume methods, which are used in most commercial CFD packages. Close relationship with FV is very attractive for CFD practitioners, as most currently successful CFD codes are based on FV framework.

The method is implemented and tested within the LLNL's ALE3D code [ale13, ALE14]. ALE3D is a multi-physics numerical simulation software tool utilizing arbitrary Lagrangian-Eulerian (ALE) techniques. The code is written to address two-dimensional (2D) and three-dimensional (3D) physics and engineering problems using a hybrid finite element and finite volume formulation on an unstructured grid. The ALE and mesh relaxation capabilities broadens the scope of applications in comparison to tools restricted to Lagrangian-only or Eulerian-only approaches, while maintaining accuracy and efficiency for large, multi-physics and complex geometry simulations. Beyond its foundation as a hydrodynamics and structures code, ALE3D has multi-physics capabilities that integrate various packages through an operator-splitting approach. Additional ALE3D features include heat conduction, chemical kinetics and species diffusion, incompressible flow, a wide range of material models, chemistry models, multi-phase flow, and magneto-hydridynamics for long (implicit) and short (explicit) time-scale applications.

The rest of this document is organized as following. We start with the description of governing equations, in Chapter 2. RDG with orthogonal basis functions is introduced in Chapter 3. Fully-implicit time discretization and related algorithms (Newton-Krylov solver, preconditioning) are discussed in Chapter 4. Extensive numerical testing of the method is performed in Chapter 5, which includes discussion of m-consistency, Section 5.1, convergence study in both time and space, for non-linear heat conduction and fluid dynamics solvers, Sections 5.2 and 5.5, mesh imprint effects, Section 5.3, modeling of phase change, Section 5.4, and modeling of flows with vortical structures, Sections 5.6 and 5.7. Finally, concluding remarks are given in Chapter 6.

This Page is Intentionally Left Blank

Chapter 2

Governing Equations

In this chapter, we present the governing equations for applications of interest. These equations can be written in the following abstract form:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial}{\partial x_j}\left(\mathbf{F}_j - \mathbf{D}_j\right) = \mathbf{S} \tag{2.1}$$

where t, $\mathbf{x} = x_j = (x, y, z)$, $\mathbf{U}$, $\mathbf{F}$, $\mathbf{D}$ and $\mathbf{S}$ are time, Cartesian coordinates, the vectors of conservative variables, hyperbolic flux, diffusion flux and source vectors, correspondingly. We also introduce a vector of "primitive" variables, $\mathbf{V}$, which is generally different from $\mathbf{U}$, and chosen based on the "better system conditioning" considerations.

2.1 Heat Conduction Model

2.1.1 Conservation of energy

Heat conduction model is based on the *conservation of total energy*,

$$\mathbf{U} = [E] \tag{2.2}$$

$$\begin{aligned} E &= \rho e \\ e &= \mathfrak{u} + \frac{\mathbf{v}^2}{2} \end{aligned} \tag{2.3}$$

where E, e, ρ, $\mathfrak{u}$ and $\mathbf{v} = v_j = (u, v, w)$ are the total energy, specific total energy, density, specific internal energy and material velocity vector, respectively.

Hyperbolic flux is defined as

$$\mathbf{F}_j = \left[v_j \left(E + P\right)\right] \tag{2.4}$$

where P is thermodynamic pressure.

Diffusion flux is

$$\mathbf{D}_j = \left[-q_j + v_k \tau_{jk}\right] \tag{2.5}$$

where q_j and τ_{jk} are heat flux vector and viscous stress tensor, respectively.

Finally, we consider energy source terms due to volumetric heat source, q''', and body force, f_j,

$$\mathbf{S} = \left[q''' + v_j f_j\right] \tag{2.6}$$

2.1.2 Internal energy formulation

The first option for *primitive solution variable* is the specific internal energy

$$\mathbf{V} = [\mathfrak{u}] \tag{2.7}$$

The energy equation can be re-written as

$$\frac{\partial}{\partial t}\left(\rho \mathfrak{u}\right) + \cancelto{0}{\frac{\partial}{\partial x_j}\left(\rho \mathfrak{u} v_j\right)} = -\frac{\partial q_j}{\partial x_j} - P\cancelto{0}{\frac{\partial v_j}{\partial x_j}} + \tau_{kj}\cancelto{0}{\frac{\partial v_k}{\partial xj}} + q''' \tag{2.8}$$

where we ignored material motion, body forces and stress tensor heating effects. Thus, heat conduction model is defined as:

$$\begin{aligned} \mathbf{U} &= [\rho \mathfrak{u}] \\ \mathbf{V} &= [\mathfrak{u}] \\ \mathbf{F}_j &= [0] \\ \mathbf{D}_j &= \left[q_j\right] \\ \mathbf{S} &= [q'''] \end{aligned} \tag{2.9}$$

where density ρ is constant.

2.1.3 Enthalpy formulation

The second option for the *primitive solution variable* is the specific enthalpy

$$\mathbf{V} = [\mathfrak{h}] \tag{2.10}$$

defined as

$$\mathfrak{h} = \mathfrak{u} + \frac{P}{\rho} \tag{2.11}$$

We generally allow pressure to be a function of time,

$$P = P(t)$$

With these, heat conduction model is defined as:

$$\begin{aligned} \mathbf{U} &= [\rho\mathfrak{h} - P] \\ \mathbf{V} &= [\mathfrak{h}] \\ \mathbf{F}_j &= [0] \\ \mathbf{D}_j &= [q_j] \\ \mathbf{S} &= [q'''] \end{aligned} \tag{2.12}$$

2.1.4 Temperature formulation

The final option for the *primitive solution variable* is temperature

$$\mathbf{V} = [T] \tag{2.13}$$

defined as

$$T = \left.\frac{\partial \mathfrak{u}}{\partial \mathfrak{s}}\right|_{\rho} \tag{2.14}$$

or

$$T = \left.\frac{\partial \mathfrak{h}}{\partial \mathfrak{s}}\right|_{P} \tag{2.15}$$

where $\mathfrak{s}$ is specific entropy. We will consider *thermally perfect* materials,

$$\mathfrak{u} = \mathfrak{u}(T)$$

and

$$\mathfrak{h} = \mathfrak{h}(T)$$

Thus,

$$\mathfrak{u}(T) = \mathfrak{u}_0 + \tilde{C}_v(T)(T - T_0) \tag{2.16}$$

where T_0 and $\mathfrak{u}_0$ are some reference constants.

Specific heats are defined as

$$C_v(T) = \left.\frac{\partial \mathfrak{u}}{\partial T}\right|_\rho = \tilde{C}_v(T) + \frac{\partial \tilde{C}_v}{\partial T} T \tag{2.17}$$

and

$$C_p(T) = \left.\frac{\partial \mathfrak{h}}{\partial T}\right|_P = C_v(T) - \frac{P}{\rho^2}\frac{\partial \rho}{\partial T} \tag{2.18}$$

(see Section C.1), which are generally functions of temperature.

With these, heat conduction model is defined as:

$$\begin{aligned} \mathbf{U} &= [\rho \mathfrak{u}] \\ \mathbf{V} &= [T] \\ \mathbf{F}_j &= [0] \\ \mathbf{D}_j &= [q_j] \\ \mathbf{S} &= [q'''] \end{aligned} \tag{2.19}$$

2.1.5 Heat flux

For heat flux, we will use *Fourier law*, defined as

$$q_j = -\kappa \frac{\partial T}{\partial x_j} \tag{2.20}$$

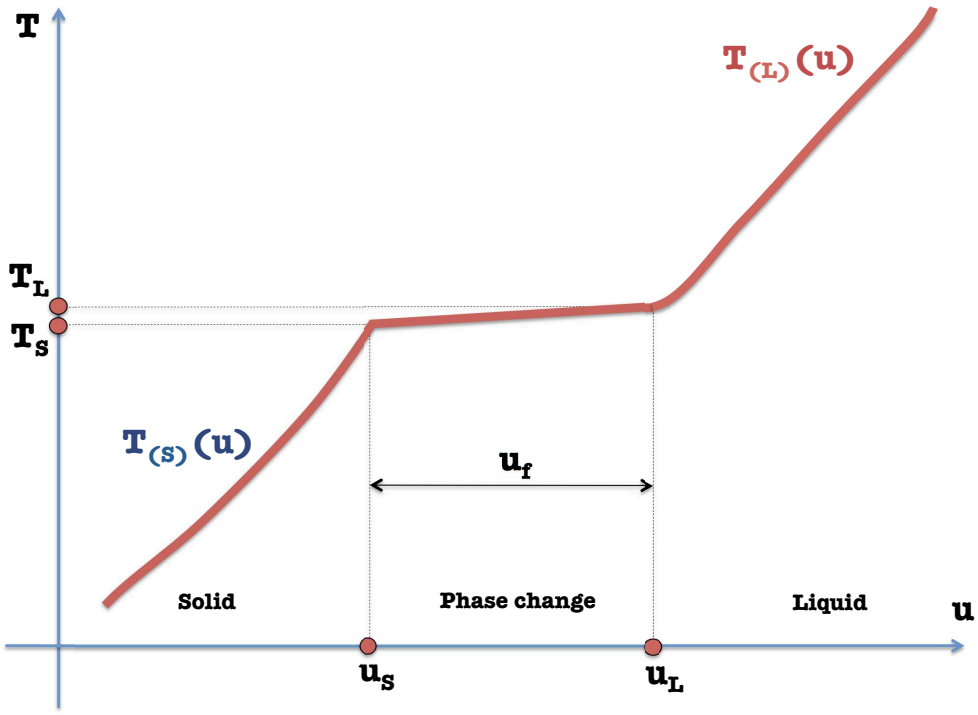


Fig. 2.1 : On melting/solidification modeling.

where κ is thermal conductivity coefficient, which is in general a function of temperature,

$$\kappa = \kappa\left(T\right)$$

2.1.6 Modeling phase change (melting/solidification)

To represent phase change due to melting/solidification, we extend the thermally perfect material model as explained in Figure 2.1. Three material state zones are introduced.

I. **Solid.**

$$\boxed{T < T_S, \;\; \mathfrak{u} < \mathfrak{u}_S}$$

where T_S and $\mathfrak{u}_S$ are *solidus* temperature and specific internal energy, re-

spectively. Material properties are

$$\begin{aligned} \mathfrak{u}_{(S)}(T) &= \mathfrak{u}_0 + \tilde{C}_{v_{(S)}}(T)(T - T_0) \\ C_{v_{(S)}}(T) &= \left.\frac{\partial \mathfrak{u}_{(S)}}{\partial T}\right|_{\rho} = \tilde{C}_{v_{(S)}}(T) + \frac{\partial \tilde{C}_{v_{(S)}}}{\partial T} T \\ \kappa_{(S)}(T) &= \kappa(T) \\ \rho_{(S)} &= const \end{aligned} \tag{2.21}$$

II. **Liquid.**

$$\boxed{T > T_L, \;\; \mathfrak{u} > \mathfrak{u}_L}$$

where T_L and $\mathfrak{u}_L$ are *liquidus* temperature and specific internal energy, respectively. Material properties are modeled as

$$\begin{aligned} \mathfrak{u}_{(L)}(T) &= \mathfrak{u}_L + \tilde{C}_{v_{(L)}}(T)(T - T_L) \\ C_{v_{(L)}}(T) &= \left.\frac{\partial \mathfrak{u}_{(L)}}{\partial T}\right|_{\rho} = \tilde{C}_{v_{(L)}}(T) + \frac{\partial \tilde{C}_{v_{(L)}}}{\partial T} T \\ \kappa_{(L)}(T) &= \kappa(T) \\ \rho_{(L)} &= const \end{aligned} \tag{2.22}$$

III. **Two-phase.**

$$\boxed{T_S \leq T \leq T_L, \;\; \mathfrak{u}_S \leq \mathfrak{u} \leq \mathfrak{u}_L}$$

We define *latent heat* as

$$\mathfrak{u}_f = \mathfrak{u}_L - \mathfrak{u}_S \tag{2.23}$$

Material properties in two-phase zone are defined by

$$\begin{aligned} T_{2\phi}(\mathfrak{u}) &= T_S + x(T_L - T_S) \\ \rho_{2\phi}(\mathfrak{u}) &= \rho_{(S)} + x\left(\rho_{(L)} - \rho_{(S)}\right) \\ C_{v_{2\phi}}(\mathfrak{u}) &= C_{v_{(S)}}(T_S) + x\left(C_{v_{(L)}}(T_L) - C_{v_{(S)}}(T_S)\right) \\ \kappa_{2\phi}(\mathfrak{u}) &= \kappa_{(S)}(T_S) + x\left(\kappa_{(L)}(T_L) - \kappa_{(S)}(T_S)\right) \end{aligned} \tag{2.24}$$

where *thermodynamic quality* is defined as

$$x = \frac{\mathfrak{u} - \mathfrak{u}_S}{\mathfrak{u}_L - \mathfrak{u}_S} \tag{2.25}$$

While most pure materials have $T_{_S} = T_{_L}$, we always allow some small difference, to enable physical thickness for melting front capturing (by smearing jumps in specific energy due to latent heat), thereby eliminating numerical degeneracy/oscillations when computing heat transfer across two-phase zone.

2.1.7 Dimensionless form

To dimensionalize energy equation (2.8), we define following scaling parameters:

$$\bar{\rho},\ \bar{T},\ \bar{L},\ \bar{t},\ \text{and}\ \bar{C}_{_v} \tag{2.26}$$

for density, temperature, length, time and heat capacity, correspondingly. With these, thermal conductivity scale is

$$\bar{\kappa} = \frac{\bar{\rho}\bar{C}_{_v}\bar{L}^{^2}}{\bar{t}} \tag{2.27}$$

specific internal energy scale is

$$\bar{\mathfrak{u}} = \bar{C}_{_v}\bar{T} \tag{2.28}$$

and dimensionless variables/parameters of the model are

$$\begin{aligned}
\hat{T} &= \tfrac{T}{\bar{T}} & \hat{\mathfrak{u}} &= \tfrac{\mathfrak{u}}{\bar{\mathfrak{u}}} \\
\hat{\mathfrak{h}} &= \tfrac{\mathfrak{h}}{\bar{\mathfrak{u}}} & \hat{\rho} &= \tfrac{\rho}{\bar{\rho}} \\
\hat{\kappa} &= \tfrac{\kappa}{\bar{\kappa}} & \hat{C}_{_v} &= \tfrac{C_{_v}}{\bar{C}_{_v}} \\
\hat{C}_{_p} &= \tfrac{C_{_p}}{\bar{C}_{_v}} & \hat{\mathbf{x}} &= \tfrac{\mathbf{x}}{\bar{L}} \\
\hat{t} &= \tfrac{t}{\bar{t}}
\end{aligned} \tag{2.29}$$

In conjunction with phase change problems (Section 2.1.6), it is useful to introduce *Stefan number* as

$$Ste = \frac{\bar{C}_{_v}\Delta T}{\bar{\mathfrak{u}}_{_f}} \tag{2.30}$$

where ΔT is some characteristic temperature difference.

2.2 Navier-Stokes Equations

2.2.1 $[\rho \mathbf{m} E]$-formulation

The governing equations of fluid dynamics are based on the conservation of mass, linear momentum and total energy,

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix} \tag{2.31}$$

The vector of hyperbolic fluxes is defined as

$$\mathbf{F}_j = \begin{bmatrix} \rho v_j \\ \rho v_j u + P\delta_{1j} \\ \rho v_j v + P\delta_{2j} \\ \rho v_j w + P\delta_{3j} \\ \rho v_j e + v_j P \end{bmatrix} \tag{2.32}$$

The vector of diffusion fluxes is defined as

$$\mathbf{D}_j = \begin{bmatrix} 0 \\ \tau_{1j} \\ \tau_{2j} \\ \tau_{3j} \\ -q_j + v_k \tau_{jk} \end{bmatrix} \tag{2.33}$$

while the source vector is

$$\mathbf{S} = \begin{bmatrix} m''' \\ f_1 \\ f_2 \\ f_3 \\ q''' + f_k v_k \end{bmatrix} \tag{2.34}$$

where m''', $\mathbf{f}$ and q''' are volumetric mass source, body force, and volumetric energy source, respectively.

This constitutes *conservative*, or $[\rho \mathbf{m} E]$-formulation (stands for mass, ρ, momentum, $\mathbf{m} = \rho \mathbf{v}$, and energy, E),

$$\mathbf{V} = \mathbf{U}$$

2.2.2 Constitutive physics

To close the model, we need to define constitutive relations for viscous stresses τ_{ij}, heat flux q_j, and pressure P (thermodynamics).

Viscous stress

Without loss of generality, we will consider here Newtonian fluids, for which viscous stress tensor is defined as [Ari, LL88]

$$\tau_{ij} = 2\mu S_{ij} + \underbrace{\left(\varsigma - \frac{2}{3}\mu\right)}_{\lambda} \partial_k v_k \delta_{ij} \tag{2.35}$$

where μ, ς and λ are the first, the second and bulk viscosity coefficients, respectively. The strain tensor is defined as

$$S_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \tag{2.36}$$

Following Stokes, the bulk and second viscosities are

$$\lambda = -\frac{2}{3}\mu$$

and

$$\varsigma = 0$$

respectively, [Ari].

Heat flux

Without loosing generality, we will use *Fourier law*, as discussed in Section 2.1.5.

Equations of state

The final constituitive relationship is due to equation of state, defining

$$\begin{aligned} \rho &= \rho\left(P, \mathfrak{u}\right) \\ T &= T\left(P, \mathfrak{u}\right) \end{aligned} \tag{2.37}$$

in either analytical or tabular form. Basics of related thermodynamics is summarized in Appendix C.

γ-law gas. One of the simplest compressible fluid equation of state is

$$P(\rho, \mathfrak{u}) = \rho(\gamma - 1)(\mathfrak{u} - \mathfrak{u}_0 + C_v T_0) - \gamma \Pi_0 \tag{2.38}$$

known as γ-law gas. Parameters $\gamma = \frac{C_P}{C_v}$, $\mathfrak{u}_0$, T_0, Π_0 and C_v are given constants (Section C.5). This EOS is appropriate for many gases in a wide range of conditions, such as Helium, Argon, and Air. With stiffening term $\gamma\Pi_0$, this equation of state can be used for liquids and solids under shocking conditions.

2-parameter fluid. Another simple fluid equation of state is

$$P(\rho) = P_0 + \rho c^2 \tag{2.39}$$

where P_0 and c are reference pressure and constant sound speed (Section C.6), respectively. We use this equation of state to represent nearly-incompressible fluids.

Incompressible fluid. Equation of state for *incompressible* fluids is

$$\rho = const \tag{2.40}$$

Therefore, thermodynamic pressure is undefined. Instead, only the gradient of the *dynamic pressure* $\nabla \bar{P}$ is of relevance, which comes out as a product of the coupled solutions for mass and momentum equations.

For thermal energy, we utilize the concept of *thermally perfect gases*, i.e.

$$\mathfrak{u}(T) = \mathfrak{u}_0 + C_v(T)(T - T_0) \tag{2.41}$$

where $\mathfrak{u}_0$ and T_0 are some reference constants. This implies that unless viscosity and mass/momentum sources are temperature dependent, the energy equation is decoupled from mass and momentum conservation equations.

2.2.3 $[P\mathbf{v}\mathfrak{u}]$-formulation

Conservative formulation in Section 2.2.1 is an excellent choice for high-speed flows, when shock wave dynamics is the dominating physics. For low-speed

flows and stiff fluids however, using conservative variables as a solution vector is detrimental for solvability of discretized equations. In particular, consider the following linearization of density around a state $(P_0, \mathfrak{u}_0)$

$$\rho(P, \mathfrak{u}) \approx \left.\frac{\partial \rho}{\partial P}\right|_{\mathfrak{u}} (P - P_0) + \left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_{P} (\mathfrak{u} - \mathfrak{u}_0)$$

For stiff fluids (such as water or liquid metals), $\left.\frac{\partial \rho}{\partial P}\right|_{\mathfrak{u}}$ and $\left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_{P}$ are huge numbers, $\sim 10^6$. Thus, small variations of density in low-Mach regimes $\rho \approx const$ cause large variations of pressure and internal energy. On the other hand, density is rather weakly sensitive (or even completely insensitive in the incompressible limit $M \to 0$) to both P and $\mathfrak{u}$. This leads to degenerate (ill-conditioned) Jacobian matrices for underlying linear algebra.

Similarly, total energy is a poor choise for low-speed flow regime, because it is very weakly dependent on kinetic energy,

$$\frac{\mathbf{v}^2}{2} \ll \mathfrak{u}$$

One of the great strengthes of the non-linear Newton-Krylov algorithm we are using (Section 4.2) is that it does not require to solve for conservative variables. Instead, we should choose solution variables which makes solution at linear steps better conditioned.

The first option is solving for pressure (P), velocity ($\mathbf{v}$) and specific internal energy ($\mathfrak{u}$), referred to as $[P\mathbf{v}\mathfrak{u}]$-formulation. Thus,

$$\mathbf{V} = \begin{bmatrix} P \\ u \\ v \\ w \\ \mathfrak{u} \end{bmatrix} \tag{2.42}$$

While replacing momentum vector $\mathbf{m}$ by velocity vector $\mathbf{v}$ does not impact significantly solvability of linear algebra, velocity is more natural (convenient, primitive) variable, offering some advantages in initialization and boundary condition treatments.

2.2.4 $[P\mathbf{v}\mathfrak{h}]$-formulation

Another useful primitive variable set is $[P\mathbf{v}\mathfrak{h}]$,

$$\mathbf{V} = \begin{bmatrix} P \\ u \\ v \\ w \\ \mathfrak{h} \end{bmatrix} \tag{2.43}$$

Using specific enthalpy $\mathfrak{h}$ instead of specific internal energy $\mathfrak{u}$ is advantageous for applications with phase change (see discussion in Section 2.3), as it is directly related to latent heat $\mathfrak{r}$:

$$\mathfrak{r}_{_{LS}} = \mathfrak{h}_{_L} - \mathfrak{h}_{_S} \tag{2.44}$$

for melting/solification, and

$$\mathfrak{r}_{_{lv}} = \mathfrak{h}_{_l} - \mathfrak{h}_{_v} \tag{2.45}$$

for evaporation/condensation. In eqs.(2.44) and (2.45), $\mathfrak{h}_{_S}$, $\mathfrak{h}_{_L}$, $\mathfrak{h}_{_l}$ and $\mathfrak{h}_{_v}$ are saturation specific enthalpies at solid, liquid and vapour states, respectively.

2.2.5 $[P\mathbf{v}T]$-formulation

Our final choise for the solution vector is $[P\mathbf{v}T]$

$$\mathbf{V} = \begin{bmatrix} P \\ u \\ v \\ w \\ T \end{bmatrix} \tag{2.46}$$

Temperature is a convenient natural (primitive) variable, which is usefull when setting initial and boundary conditions.

2.2.6 Simplifications

Boussinesq approximation for buoyancy

In fully-compressible formulation, the gravity term is usually represented as

$$\mathbf{f} = \rho\mathbf{g}$$

where $\mathbf{g}$ is a gravity vector. In the nearly-incompressible limit, this term can be replaced by *Boussinesq* approximation,

$$\mathbf{f} = \rho_0 \mathbf{g} \beta \left(T - T_0 \right) \tag{2.47}$$

where ρ_0 and T_0 are reference density and temperature, while β is the given coefficient of thermal expansion.

Energy conservation

Another simplification can be made in the limit of (nearly-) incompressible fluids, in which we can solved for *conservation of internal energy*,

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \mathfrak{u} \end{bmatrix} \tag{2.48}$$

$$\mathbf{F}_j = \begin{bmatrix} \rho v_j \\ \rho v_j u + P\delta_{1j} \\ \rho v_j v + P\delta_{2j} \\ \rho v_j w + P\delta_{3j} \\ \rho v_j \mathfrak{u} \end{bmatrix} \tag{2.49}$$

$$\mathbf{D}_j = \begin{bmatrix} 0 \\ \tau_{1j} \\ \tau_{2j} \\ \tau_{3j} \\ -q_j \end{bmatrix} \tag{2.50}$$

and

$$\mathbf{S} = \begin{bmatrix} m''' \\ f_1 \\ f_2 \\ f_3 \\ q''' \end{bmatrix} \tag{2.51}$$

Note that for the solution vector $\mathbf{V}$, one can use either eq.(2.42), eq.(2.43) or eq.(2.46). We usually use this energy conservation form in conjunction with the 2-parameter fluid EOS, Section C.6, in the low Mach number (nearly-incompressible) regime.

2.2.7 Dimensionless forms

To dimensionalize compressible Navier-Stokes equations, we have to define four scaling parameters:

$$\bar{\rho},\ \bar{L},\ \begin{bmatrix} \bar{P} \\ \bar{v} \\ \bar{f} \\ \bar{t} \end{bmatrix},\ \bar{T} \tag{2.52}$$

where we can chose either $\bar{P}$, $\bar{v}$, $\bar{f}$ or $\bar{t}$, which will be denoted as *pressure-*, *velocity-*, *body-force-* or *time-* scaling, correspondingly. The rest scaling parameters can be derived from these chosen four, using

$$\begin{aligned}
\bar{v} &= \tfrac{\bar{L}}{\bar{t}} & \bar{P} &= \bar{\rho}\bar{v}^2 & \bar{\mu} &= \tfrac{\bar{L}^2}{\bar{\rho}\bar{t}} \\
\bar{\kappa} &= \tfrac{\bar{L}\bar{\rho}\bar{v}^3}{\bar{T}} & \bar{\mathfrak{u}} &= \bar{v}^2 = \tfrac{\bar{P}}{\bar{\rho}} & \bar{C}_v &= \tfrac{\bar{v}^2}{\bar{T}} \\
\bar{f} &= \tfrac{\bar{L}}{\bar{t}^2} = \tfrac{\bar{v}^2}{\bar{L}}
\end{aligned} \tag{2.53}$$

Thus, dimensionless variables/parameters of the model are

$$\begin{aligned}
\hat{P} &= \tfrac{P}{\bar{P}} & \hat{T} &= \tfrac{T}{\bar{T}} & \hat{\mathbf{v}} &= \tfrac{\mathbf{v}}{\bar{v}} \\
\hat{\mathfrak{u}} &= \tfrac{\mathfrak{u}}{\bar{\mathfrak{u}}} & \hat{\mathfrak{h}} &= \tfrac{\mathfrak{h}}{\bar{\mathfrak{u}}} & \hat{\rho} &= \tfrac{\rho}{\bar{\rho}} \\
\hat{\kappa} &= \tfrac{\kappa}{\bar{\kappa}} & \hat{\mu} &= \tfrac{\mu}{\bar{\mu}} & \hat{C}_v &= \tfrac{C_v}{\bar{C}_v} \\
\hat{C}_p &= \tfrac{C_p}{\bar{C}_v} & \hat{\mathbf{f}} &= \tfrac{\mathbf{f}}{\bar{f}} & \hat{\mathbf{x}} &= \tfrac{\mathbf{x}}{\bar{L}} \\
\hat{t} &= \tfrac{t}{\bar{t}}
\end{aligned} \tag{2.54}$$

Reynolds number (velocity) scaling

Choosing velocity scale $\bar{v}$, the governing equations can be written in the following dimensionless form:

Mass conservation

$$\partial_{\hat{t}}\hat{\rho} + \partial_{\hat{j}}\left(\hat{\rho}\hat{\mathbf{v}}_j\right) = 0 \tag{2.55}$$

Momentum conservation

$$\partial_{\hat{t}}\hat{\rho}\hat{\mathbf{v}}_i + \partial_{\hat{j}}\left(\hat{\rho}\hat{\mathbf{v}}_i\hat{\mathbf{v}}_j + \hat{P}\right) = \partial_{\hat{j}}\left(\tfrac{1}{Re}\partial_{\hat{j}}\hat{\mathbf{v}}_i\right) + \hat{\rho}\hat{f}_i \tag{2.56}$$

and

Energy conservation

$$\partial_{\hat{t}}\hat{\rho}\hat{\mathbf{u}} + \partial_{\hat{j}}\left(\hat{\rho}\hat{\mathbf{u}}\hat{\mathbf{v}}_j\right) = \partial_{\hat{j}}\left(\tfrac{1}{RePr}\partial_{\hat{j}}\hat{T}\right) \tag{2.57}$$

where we defined the *Reynolds* and *Prandtl* numbers as

$$Re = \frac{\bar{L}\bar{v}}{\bar{\nu}} \tag{2.58}$$

and

$$Pr = \frac{\bar{\nu}}{\bar{\alpha}} \tag{2.59}$$

respectively. Kinematic viscosity and thermal diffusivity are

$$\bar{\nu} = \frac{\bar{\mu}}{\bar{\rho}}$$

and

$$\bar{\alpha} = \frac{\bar{\kappa}}{\bar{\rho}\bar{C}_v}$$

correspondingly.

Grashof number (buoyancy) scaling

Gravity scaling of governing equations is useful for natural convection flows. In this case, velocity scale is defined as

$$\bar{v} = \sqrt{\bar{g}\bar{\beta}\bar{L}\Delta T}$$

where

$$\Delta T = T - T_{\text{ref}}$$

Thus, momentum and energy conservation equations are

Momentum conservation

$$\partial_{\hat{t}} \hat{\rho} \hat{\mathbf{v}}_i + \partial_{\hat{j}} \left(\hat{\rho} \hat{\mathbf{v}}_i \hat{\mathbf{v}}_j + \hat{P} \right) = \partial_{\hat{j}} \left(\frac{1}{\sqrt{Gr}} \partial_{\hat{j}} \hat{\mathbf{v}}_i \right) + \hat{\rho} \hat{\beta} \Delta \hat{T} \hat{g}_i \tag{2.60}$$

and

Energy conservation

$$\partial_{\hat{t}} \hat{\rho} \hat{\mathfrak{u}} + \partial_{\hat{j}} \left(\hat{\rho} \hat{\mathfrak{u}} \hat{\mathbf{v}}_j \right) = \partial_{\hat{j}} \left(\frac{1}{\sqrt{Gr} Pr} \partial_{\hat{j}} \hat{T} \right) \tag{2.61}$$

correspondingly. In these equations, we introduced the *Grashof* number, defined as

$$Gr = \frac{\bar{g} \bar{\beta} \Delta T \bar{L}^3}{\bar{\nu}^2} = \frac{\bar{v}^2 \bar{L}^2}{\bar{\nu}^2} = Re^2 \tag{2.62}$$

Another useful dimensionless number is the *Rayleigh* number,

$$Ra = \frac{\bar{g} \bar{\beta} \Delta T \bar{L}^3}{\bar{\nu} \bar{\alpha}} = Gr \; Pr \tag{2.63}$$

2.3 Fluid-Solid Multiphase Flows

2.3.1 Latent heat

To represent phase change within the scope of the thermal Navier-Stokes equations (2.1) and (2.48)-(2.51), we use an *energy-based* (*homogeneous thermal equilibrium*) approach. Latent heat is incorporated into the thermal model $\mathfrak{u}(T)$, as described in Section 2.1.6. As the materials of practical interests for AM are alloys, we always have a transitional two-phase region, between solidus and liquidus. Thermal conductivity in this transitional region is represented as

$$\kappa(\mathfrak{u}) = \begin{cases} \kappa_S & \text{if } \mathfrak{u} < \mathfrak{u}_S \\ \kappa_L + \frac{\kappa_S - \kappa_L}{2} (1 + \cos(x\pi)) & \text{if } \mathfrak{u}_S \leq \mathfrak{u} \leq \mathfrak{u}_L \\ \kappa_L & \text{otherwise} \end{cases} \tag{2.64}$$

where κ_L, κ_S and x are the thermal conductivity of liquid, the thermal conductivity of solid, and thermodynamic quality, respectively.

To model the strength of the solid material, in both solid state and transitional two-phase "mushy" region, a number of approaches are available.

2.3.2 Darcy law approach

Voller and Prakash [VP87] used the following *Darcy law approach* in the "mushy" zone. Darcy law is defined as

$$\mathbf{v} = -\frac{\mathcal{K}}{\mu}\nabla P \tag{2.65}$$

where $\mathcal{K}$ is the permeability, being modeled as a function of porosity λ. Within the scope of the present study, the porosity can be approximated as

$$\lambda = 1 - \phi_{_s} = x$$

where $\phi_{_s}$ is the local solid fraction, while x is the thermodynamic quality, eq.(2.25). As the porosity decreases, both the permeability and the superficial velocity diminish as well, down to the limiting value of zero, at the completely solid state. Within the V&P model, this behavior can be represented by adding the following body forces to momentum equations:

$$\mathbf{f}_{_D} = \mathcal{A}\mathbf{v} \tag{2.66}$$

where $\mathcal{A}$ increases from zero to a very large number as the local solid fraction varies from 0 (in liquid state) to 1 (in completely solid state). In [VP87], it is modeled as

$$\mathcal{A} = -\mathcal{C}\frac{(1-\lambda)^2}{\lambda^3 + \epsilon} \tag{2.67}$$

based on the Carman-Kozeny equation. The value $\mathcal{C}$ depends on the morphology of the porous media. In [VP87], it was set to $1.6 \cdot 10^3$. Parameter ϵ is introduced to avoid division by zero, and it was set to 10^{-3}.

From the point of view numerics, this model introduces a strong source-term coupling of energy and momentum equations, akin to the Boussinesq gravity term. This requires an algorithm which enables tight coupling of momentum and energy equations. SIMPLE-like algorithms (like those used in [VP87]) are usually organized with outer Picard iterations, to account for this non-linear coupling. In [DBNS96], we used this model implemented in the commercial CFD package CFX (currently, a part of ANSYS). To keep non-disconverging solutions for internally-heated melt pools, we needed to employ hundreds of outer (Picard) iterations.

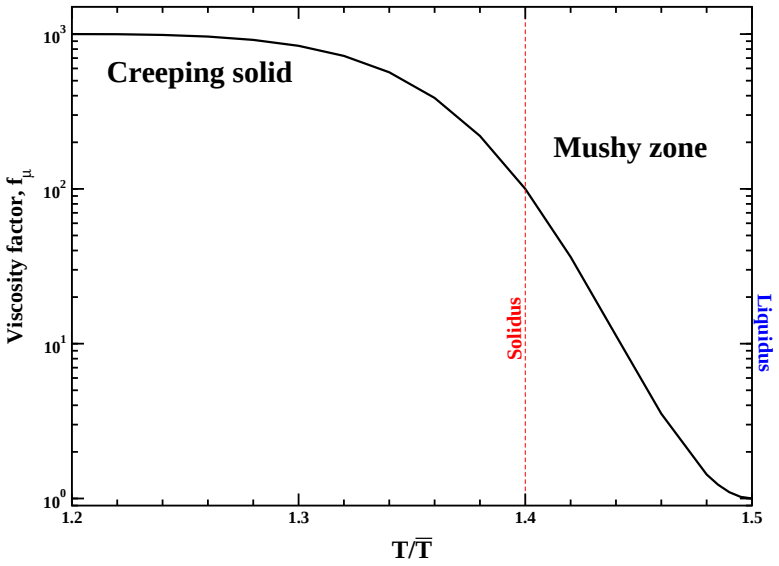


Fig. 2.2 : An example of viscosity factor, as a function of temperature.

2.3.3 Viscosity-based approach

In the present study, we use the following empirical *viscosity-based* model. Material strength is represented as a linear function of strain tensor. Thus, it is a simple extension of the viscous stress tensor in fluid, where the viscosity is set to be a function of the specific internal energy (or temperature). We use the following functional form for dynamic viscosity:

$$\mu\left(T\right)=\begin{cases}\mu_{S}^{\star} & \text{if } T<T^{\star}\\ \mu_{L}f_{\mu}\left(T\right) & \text{if } T^{\star}\leq T\leq T_{L}\\ \mu_{L} & \text{otherwise}\end{cases} \tag{2.68}$$

where $f_{\mu}\left(T\right)$ is the viscosity factor, smoothly varied from 1 to a large number, $\frac{\mu_{S}^{\star}}{\mu_{L}}$, in the range of temperatures from liquidus T_{L} down to $T^{\star}$. This model enables a smooth transition of the effective viscosity of the media from the dynamic viscosity of liquid (μ_{L}) to the very large "solid-state" viscosity ($\mu_{S}^{\star}$), which inhibits material deformation. In the present study, we defined the viscosity factor as

$$f_{\mu}\left(T\right)=10^{\varphi(T)} \tag{2.69}$$

where

$$\varphi(T) = -(a_0 - 4a_1)\psi(T) + 2(a_0 - 2a_1)\psi(T)^2 \tag{2.70}$$

$$\psi(T) = \frac{1}{2}(1 + \cos(\pi y(T))) \tag{2.71}$$

$$y(T) = b_0 + b_1\hat{T} + b_2\hat{T}^2 \tag{2.72}$$

$$a_0 = \log_{10}(\alpha f_S) \quad \text{and} \quad a_1 = \log_{10}(f_S) \tag{2.73}$$

$$\begin{aligned} b_0 &= \frac{\hat{T}^\star(\hat{T}_L(\hat{T}^\star - \hat{T}_L) - 2\hat{T}_S(\hat{T}^\star - \hat{T}_S))}{d} \\ b_1 &= \frac{\hat{T}_\star^2 + \hat{T}_L^2 - 2\hat{T}_S^2}{d} \\ b_2 &= -\frac{\hat{T}^\star + \hat{T}^L - 2\hat{T}^S}{d} \\ d &= 2\left(\hat{T}^\star - \hat{T}^L\right)\left(\hat{T}^\star - \hat{T}^S\right)\left(\hat{T}^L - \hat{T}^S\right) \\ \hat{T}^\star &= \hat{T}^L - \omega\left(\hat{T}^L - \hat{T}^S\right) \end{aligned} \tag{2.74}$$

and temperature $\hat{T}$ is dimensionalized as described in eq.(2.54). There are three input parameters for this model, i.e. f_S (viscosity factor at solidus), α (defining the limiting "solid-state" viscosity as $\mu_S^\star = \alpha f_S \mu_L$) and ω (defining the thickness of the "creeping solid" state). An example of the $f_\mu(T)$ for $\hat{T}_L = 1.5$, $\hat{T}_S = 1.4$, $f_S = 10^2$, $\alpha = 10$ and $\omega = 2$ is shown in Figure 2.2.

Similarly to the Darcy law based material strength, this model introduces a strong coupling of energy and momentum equation. However, this non-linearity is in the parabolic operator, which is rather difficult to treat numerically. We demonstrate performance of our solver for this challenging non-linear coupling in Section 5.10.

This Page is Intentionally Left Blank

Chapter 3

Space Discretization using Orthogonal RDG

In this chapter, we describe our space discretization method. We start with a brief introduction of the *Discontinuous Galerkin* (DG) method, Sections 3.1 and 3.2, followed by the description of the *recovery and reconstructed* DG (RDG) methods, and then by the introduction of the RDG based on *orthogonal* basis functions, in Section 3.4.

3.1 Method of Mean Weighted Residuals

We are concerned with the solution of general partial differential equations of type eq.(2.1), using the *Method of Mean Weighted Residuals* (MWR) [Fle91]. The solution

$$\mathbf{U} = \mathbf{U}\left(t, \mathbf{x}\right)$$

for equation (2.1) is assumed to be well approximated by a function of a particular form having *a finite set of degrees of freedom* (DoFs),

$$\mathcal{U}_i, \quad i = 0, ..., N-1$$

where N is the total number of DoFs.

We are looking for minimization of the following residue function,

$$\mathbf{R}\left(t, \mathbf{x}, \mathbf{U}, \frac{\partial \mathbf{U}}{\partial t}, \frac{\partial \mathbf{U}}{\partial x_j}, ..., \frac{\partial^n \mathbf{U}}{\partial x_j^n}\right) \equiv \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial}{\partial x_j}\left(\mathbf{F}_j - \mathbf{D}_j\right) - \mathbf{S} \rightsquigarrow 0 \qquad (3.1)$$

in order to find corresponding values of $\mathcal{U}_i$. Note, that, in general, residue function is non-linear relative to $\mathcal{U}_i$.

Every method of mean weighted residuals involves some *"basis"* and *"test"* functions that shall be denoted by $\boldsymbol{\mathcal{W}}$ and $\boldsymbol{\mathcal{B}}$, respectively.

The method of mean weighted residuals solves

$$\mathbf{R}\left(t, \mathbf{x}, \mathbf{U}, \frac{\partial \mathbf{U}}{\partial t}, \frac{\partial \mathbf{U}}{\partial x_j}, ..., \frac{\partial^n \mathbf{U}}{\partial x_j^n}\right) = 0$$

by imposing that the DoFs $\mathcal{U}_i$ are such that

$$\left(\mathbf{R}\left(t, \mathbf{x}, \mathbf{U}, \frac{\partial \mathbf{U}}{\partial t}, \frac{\partial \mathbf{U}}{\partial x_j}, ..., \frac{\partial^n \mathbf{U}}{\partial x_j^n}\right), \mathcal{W}_i\right) = 0 \tag{3.2}$$

are satisfied. The notion (f, g) is the *standard function inner product*.

The *Galerkin* method is a particular subclass of MWR, with test functions being simple functions of basis functions,

$$\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{W}}(\boldsymbol{\mathcal{B}})$$

The most commonly used form is

$$\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{B}}$$

The cases of $\boldsymbol{\mathcal{B}} \neq \boldsymbol{\mathcal{W}}$, are often referred to as *Petrov-Galerkin* methods.

3.2 Discontinuous Galerkin (DG)

Discontinuous Galerkin (DG) methods originate from the early work by Reed and Hill [RH73] for the solution of neutron transport equations. In late 1980s, DG evolved to solve fluid dynamics equations, [CS89, CLS89, CHS90, Coc89, Li05, LBL08, LBL06].

3.2.1 Formulation

DG methods are a particular sub-class of MWR methods, with piece-wise discontinuous basis functions. We assumed that the computational domain Ω is subdivided into a collection of non-overlapping elements, Ω_e. Without loss of generality, we will consider here linear QUAD4 (4-node) and HEX8 (8-node) elements[1].

Next, let us introduce the following broken Sobolev space $\mathbb{V}_h^p$

$$\mathbb{V}_h^p = \left\{ v_h \in \left[\mathcal{L}_2\left(\Omega\right)\right]^m : v_h|_{\Omega_e} \in \left[\mathcal{V}_p^m\right] \forall \Omega_e \in \Omega \right\} \tag{3.3}$$

which consists of discontinuous vector-values polynomial functions of degree p, and where m is the dimension of the unknown vector and $\mathcal{V}_p$ is the space of all polynomials of degree $\leq p$. To formulate basic DG method, we introduce the following weak formulation, which is obtained by multiplying eq.(2.1) by a test function $\boldsymbol{\mathcal{W}}_h$, integrating over an element Ω_e, and then performing an integration by parts,

$$\begin{aligned} \mathbf{R}_h\left(\mathbf{U}_h\right) = \frac{\partial}{\partial t} \int_{\Omega_e} \mathbf{U}_h \boldsymbol{\mathcal{W}}_h d\Omega + \int_{\Gamma_e} \left(\mathbf{F}_j\left(\mathbf{U}_h\right) - \mathbf{D}_j\left(\mathbf{U}_h\right)\right) n_j \boldsymbol{\mathcal{W}}_h d\Gamma - \\ - \int_{\Omega_e} \left[\left(\mathbf{F}_j\left(\mathbf{U}_h\right) - \mathbf{D}_j\left(\mathbf{U}_h\right)\right) \frac{\partial \boldsymbol{\mathcal{W}}_h}{\partial x_k} + \mathbf{S}\left(\mathbf{U}_h\right) \boldsymbol{\mathcal{W}}_h \right] d\Omega, \quad \forall \boldsymbol{\mathcal{W}}_h \in \mathbb{V}_h^p \end{aligned} \tag{3.4}$$

where $\mathbf{U}_h$ and $\boldsymbol{\mathcal{W}}_h$ are represented by piecewise-polynomial functions of degrees p, which are discontinuous between the cell interfaces, and $\mathbf{n} = n_j$ denotes the unit outward normal vector to the element face Γ_e (i.e., the boundary of Ω_e). This formulation is called *discontinuous Galerkin method of degree* p, and denoted as DG(P$_p$).

Since the numerical solution $\mathbf{U}_h$ is discontinuous across element interfaces, the numerical hyperbolic fluxes are not uniquely defined. The hyperbolic flux function $\mathbf{F}_j\left(\mathbf{U}_h\right) n_j$ appearing in the face integral term of eq.(3.4) is replaced by a numerical Riemann flux function, $\mathbf{H}_j\left(\mathbf{U}_h^{\mathrm{L}}, \mathbf{U}_h^{\mathrm{R}}\right) n_j$, which is computed by some (approximate) Riemann solver. Here, $\mathbf{U}_h^{\mathrm{L}}$ and $\mathbf{U}_h^{\mathrm{R}}$ are the conservative state vectors at the left and right side of the element boundary. In this sense, discontinuous Galerkin formulation is very similar to finite volume schemes. In fact,

[1] Extension to quadratic QUAD8 and HEX20 elements, as well as to triangle and tetrahedral elements is straightforward.

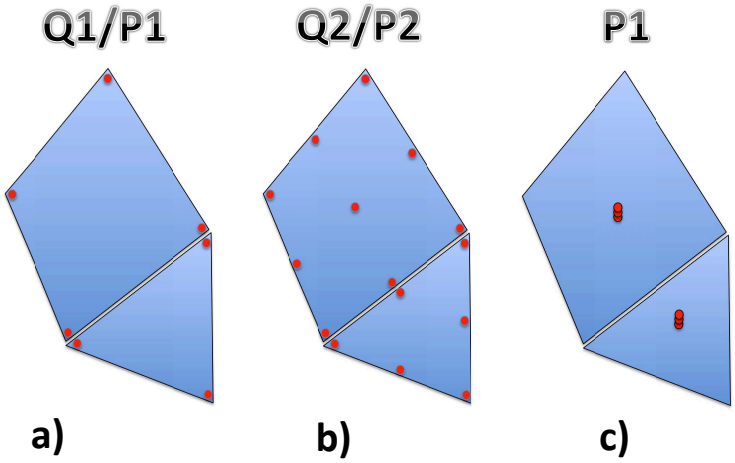


Fig. 3.1 : Representation of polynomial solutions using finite element shape functions in nodal DG (a) and (b), and using Taylor based basis functions in modal DG (c).

the classical first-order cell-centered finite volume scheme is exacly DG(P_0) - i.e. discontinuous Galerkin method using a piecewise-constant basis function. Therefore, the DG(P_k) methods with $k > 0$ can be regarded as a natural generalization of finite volume methods to higher order. By simply increasing the degree p of the polynomials, the DG methods of corresponding higher order are obtained. The domain and boundary integrals in eq.(3.4) are usually calculated using Gauss quadrature formulas. The number of quadrature points used is chosen to integrate exactly polynomials of $(2p)^{\text{th}}$ and $(2p+1)^{\text{th}}$ order for volume and surface inner products in the reference element.

3.2.2 Basis and test functions

In the most common form of DGM, numerical polynomial solutions $\mathbf{U}_h$ in each element are expressed using either standard Lagrange finite element or hierarchical node-based basis functions [Li05], as following

$$\mathbf{U}_h(\mathbf{x}) = \sum_{k=0}^{K-1} \mathbf{U}_e^{(k)}(t)\, \mathcal{B}_{(k)}(\mathbf{x}) \tag{3.5}$$

where $\mathcal{B}_{(k)}$ are the finite element basis functions. Thus, the unknowns to be solved for are the solution variables at the nodes of element, therefore the method is called *nodal* DGM, Figure 3.1(a) and (b). Also, as in the classical Galerkin formulation, the test functions coincide with the basis functions,

$$\mathcal{W}_k = \mathcal{B}_k$$

In our previous work [LBL08], [LBL06], [LLNM10b], [LLNC11], [LLN12], [LXL+12], [LXS+13], [XLFN14], [XLN14] [LLNM09], [LLNM10c], we utilized ***Taylor-series-based basis functions***, i.e., in 3D:

$$\boldsymbol{\mathcal{B}}(\mathbf{x}) = \left\{ \begin{array}{l} 1, \underbrace{\frac{x-x_c}{\Delta x}}_{\mathcal{B}_{(1)}}, \underbrace{\frac{y-y_c}{\Delta y}}_{\mathcal{B}_{(2)}}, \underbrace{\frac{z-z_c}{\Delta z}}_{\mathcal{B}_{(3)}}, \underbrace{\frac{\mathcal{B}_{(1)}^2}{2} - \frac{1}{\Omega_e}\int\limits_{\Omega_e} \frac{\mathcal{B}_{(1)}^2}{2} d\Omega}_{\mathcal{B}_{(4)}}, \\ \underbrace{\frac{\mathcal{B}_{(2)}^2}{2} - \frac{1}{\Omega_e}\int\limits_{\Omega_e} \frac{\mathcal{B}_{(2)}^2}{2} d\Omega}_{\mathcal{B}_{(5)}}, \underbrace{\frac{\mathcal{B}_{(3)}^2}{2} - \frac{1}{\Omega_e}\int\limits_{\Omega_e} \frac{\mathcal{B}_{(3)}^2}{2} d\Omega}_{\mathcal{B}_{(6)}}, \\ \underbrace{\mathcal{B}_{(1)}\mathcal{B}_{(2)} - \frac{1}{\Omega_e}\int\limits_{\Omega_e} \mathcal{B}_{(1)}\mathcal{B}_{(2)} d\Omega}_{\mathcal{B}_{(7)}}, \underbrace{\mathcal{B}_{(1)}\mathcal{B}_{(3)} - \frac{1}{\Omega_e}\int\limits_{\Omega_e} \mathcal{B}_{(1)}\mathcal{B}_{(3)} d\Omega}_{\mathcal{B}_{(8)}}, \\ \underbrace{\mathcal{B}_{(2)}\mathcal{B}_{(3)} - \frac{1}{\Omega_e}\int\limits_{\Omega_e} \mathcal{B}_{(2)}\mathcal{B}_{(3)} d\Omega}_{\mathcal{B}_{(7)}}, \ldots \end{array} \right\} \tag{3.6}$$

where

$$\Delta x = \frac{x_{\max} - x_{\min}}{2}, \quad \Delta y = \frac{y_{\max} - y_{\min}}{2} \text{ and } \Delta z = \frac{z_{\max} - z_{\min}}{2}$$

and $x_{\max}$, $x_{\min}$, $y_{\max}$, $y_{\min}$, $z_{\max}$ and $z_{\min}$ are the maximum and minimum coordinates in the cell Ω_e in x-, y- and z-directions, respectively. This formulation is called *modal* DGM, Figure 3.1(c), and it has a number of useful features:

+ First, DoFs are functions of cell-averaged variables and their spatial derivatives, which is very convenient for discretization of diffusion and parabolic operators.

+ Second, these basic functions are hierarchical, which facilitates p-refinement.

+ Third, these basic functions decouple the solved for degrees of freedom from the element geometry, i.e. the same basis functions can be used for any shapes of elements: linear/quadratic tetrahedron, pyramid, prism and hexahedron in 3D, and triangles or quads in 2D. This feature is particularly important for implementation on arbitrary (hybrid) meshes, with h-refinement.

+ Finally, the zeroth-order DoF is decoupled from the rest (higher order) DoFs,

$$\int_{\Omega_e} \mathcal{B}_{(0)} \mathcal{B}_{(k)} d\Omega = 0, \quad k > 0$$

and its evolution equation concides almost exactly with the cell-centered finite volume formulation.

The obvious drawback of Taylor basis functions (as well as those used in nodal DG) – ***they are not orthogonal***. Thus, eq.(3.4) becomes

$$\mathbf{R}_h\left(\mathbf{U}_h\right) = \begin{cases} \frac{\partial}{\partial t} \int\limits_{\Omega_e} \mathbf{U}_e^{(0)} d\Omega + \int\limits_{\Gamma_e} \left[\mathbf{F}_j\left(\mathbf{U}_h\right) - \mathbf{D}_j\left(\mathbf{U}_h\right)\right] n_j d\Gamma - \int\limits_{\Omega_e} \mathbf{S}\left(\mathbf{U}_h\right) d\Omega \\ \mathbb{M}_{\mathrm{K}\times\mathrm{K}} \frac{\partial}{\partial t} \int\limits_{\Omega_e} \begin{bmatrix} \mathbf{U}_e^{(1)} \\ \mathbf{U}_e^{(2)} \\ \cdots \\ \mathbf{U}_e^{(\mathrm{K})} \end{bmatrix} d\Omega + \mathbb{S}_{\mathrm{K}\times 1} \end{cases} \tag{3.7}$$

with non-diagonal mass matrix $\mathbb{M}_{\mathrm{K}\times\mathrm{K}}$, where K+1 is the total number of DoF per element, per variable. This is rather damaging for condition number of high order ($p > 1$) versions. This deficiency is especially evident when applied within the fully-implicit time discretization context.

3.3 Recontruction and Recovery DG

As discussed above, the discontinuous Galerkin method provides an elegant way to build high-order space discretization on unstructured meshes by simply adding additional degrees of freedom per element, per variable. This approach is compact, i.e. DG(P_p) of any order p has the same stencil, which is very attractive feature in terms of parallelization and code design. On the other hand, the size of the solution vector is growing significantly, as more equations for DoFs must be solved for. In particular, for modal DG in 2D – the size of the solution vector for the second- DG(P_1), the third- DG(P_2) and the fourth-order DG(P_3) schemes are $\times 3$, $\times 6$ and $\times 10$ times larger, respectively, as compared to the basic DG(P_0) finite volume method. This is even worse for 3D, i.e. $\times 4$, $\times 10$ and $\times 20$ – for the DG(P_1), DG(P_2) and DG(P_3), correspondingly. Such an increase in the size of the solution vector is unfavourable in the context of implicit solvers, imposing significant memory requirements (for storage of linear algebra matrices) and adversely affecting solution scalability, as a majority of linear solvers do not scale linearly [Saa03].

There is an alternative way to increase the order of accuracy, which is a routine in finite volume methods. All practically used FV methods *reconstruct in-cell slopes* of the solution vector by means of the *solution reconstruction*. This leads to an increase of the stencil, which is a well-established acceptable strategy for the second-order FV methods. However, going beyond the second order is impractical for unstructured meshes, leading to significant complications in the code design.

3.3.1 Recovery based DG

In [NTP+08, NPM09], we explored the ideas of *in-cell recovery* to enhance the order of accuracy for DG, without increasing the size of the solution vector. We capitalized on the earlier work by van Leer and Nomura [vLN05], who suggested to use recovery in order to consistently discretize diffusion operator within the DG framework. In [vLN05], the DG's piece-wise discontinuous representation of the solution at the element interfaces was replaced by *locally recovered underlying smooth solution with sufficient fidelity*, at the union of face-neighboring elements. During the *recovery*, it is required that the solution is consistent with the underlying DG's broken Sobolev space solution representation. We denote this type of recovery as **inter-cell recovery**.

In [NTP+08], we noticed that similar strategy[2] can be used for enhancing **in-cell** solution representation, by considering an element and its local neighbors. In 1D, this leads to a family of $\mathrm{RDG}_{\mathrm{P_nP_{3n+2}}}$ schemes. Combined with inter-cell recovery, the $\mathrm{RDG}_{\mathrm{P_nP_{3n+2}}}$ offers very usefull discretization framework for modeling one-dimensional fluid flows, in nuclear safety system analysis codes [YNK+11].

Extending in-cell recovery to 2D regular meshes was explored in [NPM09]. We demonstrated feasibility of $\mathrm{RDG}_{\mathrm{P_1P_5}}$ for compressible Navier-Stokes and neutron diffusion equations. However, using recovery on arbitrary meshes is impractical, as it requires involvement of vertex neighbours. More fruitful approach was explored by Dumbster et al. [DBTM08, DZ09, Dum10], who utilized least-squares recovery, to achieve higher order representation with DG methods. The method is called $\mathrm{P_nP_m}$ – i.e., "solving for $\mathrm{P_n}$" and "reconstructing to $\mathrm{P_m}$" – a convenient way to reflect what we attempt to achieve with in-cell recovery.

Inter-cell recovery [vLN05] was further developed in several studies by van Leer, Lo and Raalte [vLLR07, vRvL08, vLL09, LvL09]. Of these, [LvL09] is of particular significance, as it presents a proof of method's global stability, by considering energy decay in time. We will discuss inter-cell recovery in more details in Section 3.4.

3.3.2 Reconstruction based DG

As we mentioned above, using recovery is not practical approach for increasing the order of the DG method, on arbitrary unstructured meshes. In [LLNM09], [LLNM10c], [LLNM10b], [LLNM10a], [LLNC11], [LLN12], [LXL+12], [LXS+13], [XLFN14] and [XLN14], we utilized the **least-squares based reconstruction** instead. In this approach, we increase the accuracy of the DG method by one order, using only face-neighboring elements. This is an important design requirement, necessary for feasibility of discretization on arbitrary (hybrid) unstructured meshes. The distinguishable feature of this $\mathrm{RDG}_{\mathrm{P_nP_{n+1}}}$ method is the use of *strong* statements, in difference to *weak* (integral) statements of the recovery schemes. This strategy greatly simplifies the algorithm, and it is along the lines of what is traditionally used for reconstruction with finite volume methods. In fact, the $\mathrm{RDG}_{\mathrm{P_0P_1}}$ is exactly what is known as the least-squares based finite vol-

[2] It is instructive to note that similar recovery strategy was used by Qui and Shu [QS03], but in different context, building WENO limiters for DG.

ume method [BJ89]. The $\mathrm{RDG}_{\mathrm{P_nP_{n+1}}}$ is designed to work with modal DG, capitalizing on the availability of the solution derivatives at element centers. Thus, the in-cell higher order DoFs are reconstructed hierarchically, on the top of the solved-for DoFs, with the requirement to be consistent with cell-centered solutions (and all available solved for derivatives) in all face-neighboring elements, satisfied in the least-squares sense. The method is constructed in conjunction with the *inter-cell reconstruction*, which considers all unions of face-neighboring elements, and imposes integral consistency of the face-centered reconstructed solution for cell-averaged quantities, and least-squares based strong statements for solved-for cell-centered solutions and all available derivatives. This combination of *in-cell* and *inter-cell reconstructions* provides a flexible discretization framework for generic PDEs of type eq.(2.1), which includes transient hyperbolic, parabolic or elliptic operators. Implementation of the solution limiting, necessary for shock dynamics and multi-material applications, is done within the hierarchical WENO concept, as demostrated in [LXL+12, LXS+13, XLN14].

It must be noted that the reconstructed $\mathrm{RDG}_{\mathrm{P_nP_{n+1}}}$ suffers from the same non-orthogonality issue, as its base modal DG – i.e., the mass matrix is non-diagonal. In the following sections, we describe the $\mathrm{RDG}_{\mathrm{P_nP_{n+1}}}$ based on orthogonal basis functions, as well as other practical implementation issues when building upon the Newton-Krylov based fully-implicit solution strategy.

3.4 RDG with orthogonal basis functions

Without loss of generality, we will consider the 4-node linear QUAD4 and the 8-node linear HEX8 elements, Figure 3.2, as required by the application code of interest.

3.4.1 Isoparametric mapping

The canonical shape functions are

$$\begin{aligned} \phi_0(\xi,\eta) &= \tfrac{1}{4}(1-\xi)(1-\eta) \qquad \phi_1(\xi,\eta) = \tfrac{1}{4}(1+\xi)(1-\eta) \\ \phi_2(\xi,\eta) &= \tfrac{1}{4}(1+\xi)(1+\eta) \qquad \phi_3(\xi,\eta) = \tfrac{1}{4}(1-\xi)(1+\eta) \end{aligned} \tag{3.8}$$

for QUAD4 elements, and

$$\begin{aligned}
&\phi_0(\xi,\eta,\zeta) = \tfrac{1}{8}(1-\xi)(1-\eta)(1-\zeta) \qquad \phi_1(\xi,\eta,\zeta) = \tfrac{1}{8}(1+\xi)(1-\eta)(1-\zeta)\\
&\phi_2(\xi,\eta,\zeta) = \tfrac{1}{8}(1+\xi)(1+\eta)(1-\zeta) \qquad \phi_3(\xi,\eta,\zeta) = \tfrac{1}{8}(1-\xi)(1+\eta)(1-\zeta)\\
&\phi_4(\xi,\eta,\zeta) = \tfrac{1}{8}(1-\xi)(1-\eta)(1+\zeta) \qquad \phi_5(\xi,\eta,\zeta) = \tfrac{1}{8}(1+\xi)(1-\eta)(1+\zeta)\\
&\phi_6(\xi,\eta,\zeta) = \tfrac{1}{8}(1+\xi)(1+\eta)(1+\zeta) \qquad \phi_7(\xi,\eta,\zeta) = \tfrac{1}{8}(1-\xi)(1+\eta)(1+\zeta)
\end{aligned} \tag{3.9}$$

for HEX8 elements. These functions are used for isoparametric mapping from the physical space $\mathbf{x} = (x, y, z)$ to the reference space $\boldsymbol{\chi} = (\xi, \eta, \zeta)$, Figure 3.2:

$$\begin{cases} x(\xi,\eta) &= \sum\limits_{n=0}^{3} x_n \phi_n(\xi,\eta) \\ y(\xi,\eta) &= \sum\limits_{n=0}^{3} y_n \phi_n(\xi,\eta) \end{cases} \tag{3.10}$$

for QUAD4, and

$$\begin{cases} x(\xi,\eta,\zeta) &= \sum\limits_{n=0}^{7} x_n \phi_n(\xi,\eta,\zeta) \\ y(\xi,\eta,\zeta) &= \sum\limits_{n=0}^{7} y_n \phi_n(\xi,\eta,\zeta) \\ z(\xi,\eta,\zeta) &= \sum\limits_{n=0}^{7} z_n \phi_n(\xi,\eta,\zeta) \end{cases} \tag{3.11}$$

for HEX8 elements.

The Jacobians of transformations $\mathbb{J}$ are defined by

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \underbrace{\begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix}}_{\mathbb{J}} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} \tag{3.12}$$

for 2D, and

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \underbrace{\begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix}}_{\mathbb{J}} \begin{bmatrix} d\xi \\ d\eta \\ d\zeta \end{bmatrix} \tag{3.13}$$

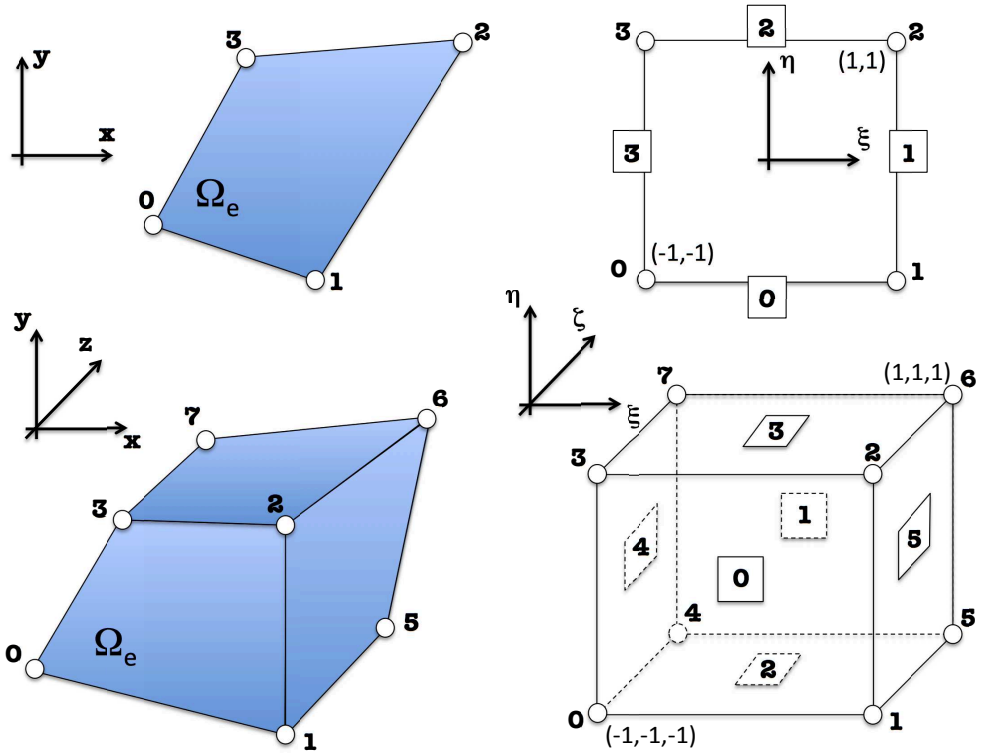


Fig. 3.2 : Element topology and isoparametric mapping for linear QUAD4 (top) and HEX8 (bottom) elements.

for 3D elements. Inverse of the Jacobian matrix is defined as

$$\mathbb{J}^{-1} = \begin{bmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{bmatrix} \tag{3.14}$$

in 2D, and

$$\mathbb{J}^{-1} = \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \tag{3.15}$$

in 3D. All these element's geometrical parameters, as well as $|\mathbb{J}|$ and partials

$$\frac{\partial \mathbb{J}^{-1}}{\partial \boldsymbol{\chi}_i}, \quad \text{and} \quad \frac{\partial^2 \mathbb{J}^{-1}}{\partial \boldsymbol{\chi}_i \boldsymbol{\chi}_j}$$

where

$$\boldsymbol{\chi} = (\xi, \eta, \zeta)$$

are needed for the high-order discretization over elements, and they can be computed from the nodal coordinates.

3.4.2 Integration

Domain integrals of a function $f(\mathbf{x})$ are computed using Gauss quadrature formulas, as

$$\begin{aligned} \iint_{\Omega_e} f(\mathbf{x})\, d\Omega &= \int_{-1}^{1}\int_{-1}^{1} f\left(x(\boldsymbol{\chi}), y(\boldsymbol{\chi})\right) \underbrace{\left|\frac{\partial(x,y)}{\partial(\xi,\eta)}\right|}_{|\mathbb{J}|} d\xi\, d\eta = \\ &= \sum_{g=0}^{N_g} \omega_g f_g |\mathbb{J}| \end{aligned} \tag{3.16}$$

for QUAD, and

$$\begin{aligned} \iiint_{\Omega_e} f(\mathbf{x})\, d\Omega &= \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f\left(x(\boldsymbol{\chi}), y(\boldsymbol{\chi}), z(\boldsymbol{\chi})\right) \underbrace{\left|\frac{\partial(x,y,z)}{\partial(\xi,\eta,\zeta)}\right|}_{|\mathbb{J}|} d\xi\, d\eta\, d\zeta = \\ &= \sum_{g=0}^{N_g} \omega_g f_g |\mathbb{J}| \end{aligned} \tag{3.17}$$

for HEX elements, where N_g, ω_g, f_g and $|\mathbb{J}|$ are the total number of Gaussian integration points, the weight of g^{th} integration point, the function evaluated at the g^{th} integration point, and the determinant of the Jacobian matrix, respectively.

3.4.3 Basis and test functions

For QUAD elements, we use the following Legendre-polynomials-based basis functions:

$$\boldsymbol{\mathcal{B}}(\xi,\eta)=\left\{\begin{array}{l} 1,\ \underbrace{\mathfrak{L}_{(1)}(\xi)}_{\mathcal{B}_{(1)}},\ \underbrace{\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(2)}},\ \underbrace{\mathfrak{L}_{(2)}(\xi)}_{\mathcal{B}_{(3)}},\ \underbrace{\mathfrak{L}_{(1)}(\xi)\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(4)}},\ \underbrace{\mathfrak{L}_{(2)}(\eta)}_{\mathcal{B}_{(5)}}, \\ \underbrace{\mathfrak{L}_{(3)}(\xi)}_{\mathcal{B}_{(6)}},\ \underbrace{\mathfrak{L}_{(2)}(\xi)\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(7)}},\ \underbrace{\mathfrak{L}_{(1)}(\xi)\mathfrak{L}_{(2)}(\eta)}_{\mathcal{B}_{(8)}},\ \underbrace{\mathfrak{L}_{(3)}(\eta)}_{\mathcal{B}_{(9)}},\ \ldots \end{array}\right\} \tag{3.18}$$

and

$$\mathbf{U}(\xi,\eta)=\sum_{k=0}^{K-1}\mathbf{U}_{(k)}\mathcal{B}_{(k)}(\xi,\eta) \tag{3.19}$$

For HEX elements, the basis functions are defined as:

$$\boldsymbol{\mathcal{B}}(\xi,\eta,\zeta)=\left\{\begin{array}{l} 1,\ \underbrace{\mathfrak{L}_{(1)}(\xi)}_{\mathcal{B}_{(1)}},\ \underbrace{\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(2)}},\ \underbrace{\mathfrak{L}_{(1)}(\zeta)}_{\mathcal{B}_{(3)}},\ \underbrace{\mathfrak{L}_{(2)}(\xi)}_{\mathcal{B}_{(4)}},\ \underbrace{\mathfrak{L}_{(2)}(\eta)}_{\mathcal{B}_{(5)}},\ \underbrace{\mathfrak{L}_{(2)}(\zeta)}_{\mathcal{B}_{(6)}}, \\ \underbrace{\mathfrak{L}_{(1)}(\xi)\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(7)}},\ \underbrace{\mathfrak{L}_{(1)}(\xi)\mathfrak{L}_{(1)}(\zeta)}_{\mathcal{B}_{(8)}},\ \underbrace{\mathfrak{L}_{(1)}(\eta)\mathfrak{L}_{(1)}(\zeta)}_{\mathcal{B}_{(9)}}, \\ \underbrace{\mathfrak{L}_{(3)}(\xi)}_{\mathcal{B}_{(10)}},\ \underbrace{\mathfrak{L}_{(3)}(\eta)}_{\mathcal{B}_{(11)}},\ \underbrace{\mathfrak{L}_{(3)}(\zeta)}_{\mathcal{B}_{(12)}},\ \underbrace{\mathfrak{L}_{(1)}(\xi)\mathfrak{L}_{(1)}(\eta)\mathfrak{L}_{(1)}(\zeta)}_{\mathcal{B}_{(13)}}, \\ \underbrace{\mathfrak{L}_{(2)}(\xi)\mathfrak{L}_{(1)}(\eta)}_{\mathcal{B}_{(14)}},\ \underbrace{\mathfrak{L}_{(1)}(\xi)\mathfrak{L}_{(2)}(\eta)}_{\mathcal{B}_{(15)}},\ \underbrace{\mathfrak{L}_{(2)}(\xi)\mathfrak{L}_{(1)}(\zeta)}_{\mathcal{B}_{(16)}}, \\ \underbrace{\mathfrak{L}_{(1)}(\xi)\mathfrak{L}_{(2)}(\zeta)}_{\mathcal{B}_{(17)}},\ \underbrace{\mathfrak{L}_{(2)}(\eta)\mathfrak{L}_{(1)}(\zeta)}_{\mathcal{B}_{(18)}},\ \underbrace{\mathfrak{L}_{(1)}(\eta)\mathfrak{L}_{(2)}(\zeta)}_{\mathcal{B}_{(19)}},\ \ldots \end{array}\right\} \tag{3.20}$$

and

$$\mathbf{U}(\xi,\eta,\zeta)=\sum_{k=0}^{K-1}\mathbf{U}_{(k)}\mathcal{B}_{(k)}(\xi,\eta,\zeta) \tag{3.21}$$

It is instructive to note that, in this formulation, the first degree of freedom is the cell-averaged quantity, similar to the Taylor basis functions formulation. Thus, this is a natural extension of the finite-volume method to high orders.

In eqs.(3.18) and (3.20), $\mathfrak{L}_{(n)}(x)$ are the *Legendre polynomials* of the order n, which can be expressed by simple monomials with multiplicative formula of the binomial coefficient:

$$\mathfrak{L}_{(n)}(x) = 2^n \cdot \sum_{k=0}^{n} x^k \begin{pmatrix} n \\ k \end{pmatrix} \begin{pmatrix} \frac{n+k-1}{2} \\ n \end{pmatrix} \tag{3.22}$$

where the binomial coefficients can be computed by

$$\begin{pmatrix} m \\ j \end{pmatrix} = \prod_{i=1}^{j} \frac{m+1-i}{i} \tag{3.23}$$

The test functions are defined as

$$\mathcal{W}_{(n)} = \frac{\mathcal{B}_{(n)}}{|\mathbb{J}|} \tag{3.24}$$

Importantly, the basis functions eqs.(3.18) and (3.20) are orthogonal relative to the test functions eq.(3.24). Thus,

$$\int_{\Omega_e} \mathcal{B}_{(n)} \mathcal{W}_{(k)} d\Omega = \frac{1}{\mathcal{A}_{(n)}} \delta_{k,n} \tag{3.25}$$

where the normalization coefficients are

$$\mathcal{A}_{(n)} = \left\{ \frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{5}{4}, \frac{9}{4}, \frac{5}{4}, \frac{7}{4}, \frac{15}{4}, \frac{15}{4}, \frac{7}{4}, \ldots \right\} \tag{3.26}$$

for QUADs, and

$$\mathcal{A}_{(n)} = \left\{ \frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{3}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}, \frac{9}{8}, \frac{9}{8}, \frac{9}{8}, \frac{7}{8}, \frac{7}{8}, \frac{7}{8}, \frac{27}{8}, \frac{15}{8}, \frac{15}{8}, \frac{15}{8}, \frac{15}{8}, \frac{15}{8}, \frac{15}{8}, \ldots \right\} \tag{3.27}$$

for HEX elements.

With these, given a function $\mathbf{U}(x, y, z)$ within an element, its n-th momentum (degree of freedom) can be computed as

$$\mathbf{U}_{(n)} = \sum_{g}^{N_g} \omega_g \mathbf{U}(\mathbf{x}_g) \mathcal{B}_{(n)}(\boldsymbol{\chi}_g) \tag{3.28}$$

where $\mathbf{x}_g = (x_g, y_g, z_g)$, $\boldsymbol{\chi}_g = (\xi_g, \eta_g, \zeta_g)$, and ω_g are the physical-space coordinate of, the reference-space coordinate of, and the weight of the g-th Gaussian quadrature point, respectively; while N_g is the total number of integration points.

For evaluation of diffusion operators and for reconstruction/recovery, it is important to have a mapping from the orthogonal basis functions eq.(3.20) to the Taylor basis functions eq.(3.6). This mapping is defined by matrices denoted as $\mathbb{L}_{2\mathrm{T}}^{(\mathrm{P_n})}$. Defining the vector of orthogonal DoFs in element as $\mathbf{U}_{(k)}$, and the vector of Taylor DoFs as $\mathbf{V}_{(k)}$, the third-order mapping $\mathbb{L}_{2\mathrm{T}}^{(\mathrm{P_2})}$ in 2D is

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \xi_{x_0} & \eta_{x_0} & 0 & 0 & 0 \\ 0 & \xi_{y_0} & \eta_{y_0} & 0 & 0 & 0 \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,4}}^{(\mathrm{P_2})} & \mathbb{L}_{2\mathrm{T}_{3,4}}^{(\mathrm{P_2})} & 3\xi_{x_0}^2 & 2\xi_{x_0}\eta_{x_0} & 3\eta_{x_0}^2 \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,5}}^{(\mathrm{P_2})} & \mathbb{L}_{2\mathrm{T}_{3,5}}^{(\mathrm{P_2})} & 3\xi_{x_0}\xi_{y_0} & \xi_{x_0}\eta_{y_0} + \xi_{y_0}\eta_{x_0} & 3\eta_{x_0}\eta_{y_0} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,6}}^{(\mathrm{P_2})} & \mathbb{L}_{2\mathrm{T}_{3,6}}^{(\mathrm{P_2})} & 3\xi_{y_0}^2 & 2\xi_{y_0}\eta_{y_0} & 3\eta_{y_0}^2 \end{bmatrix}}_{\mathbb{L}_{2\mathrm{T}}^{(\mathrm{P_2})}} \begin{bmatrix} \mathbf{U}_{(0)} \\ \mathbf{U}_{(1)} \\ \mathbf{U}_{(2)} \\ \mathbf{U}_{(3)} \\ \mathbf{U}_{(4)} \\ \mathbf{U}_{(5)} \end{bmatrix} = \begin{bmatrix} \mathbf{V}_{(0)} \\ \mathbf{V}_{(1)} \\ \mathbf{V}_{(2)} \\ \mathbf{V}_{(3)} \\ \mathbf{V}_{(4)} \\ \mathbf{V}_{(5)} \end{bmatrix} \tag{3.29}$$

where

$$\begin{aligned} \mathbb{L}_{2\mathrm{T}_{2,4}}^{(\mathrm{P_2})} &= \eta_{x_0}\partial_\eta \xi_{x_0} + \xi_{x_0}\partial_\xi \xi_{x_0} & \mathbb{L}_{2\mathrm{T}_{3,4}}^{(\mathrm{P_2})} &= \eta_{x_0}\partial_\eta \eta_{x_0} + \xi_{x_0}\partial_\xi \eta_{x_0} \\ \mathbb{L}_{2\mathrm{T}_{2,4}}^{(\mathrm{P_2})} &= \eta_{y_0}\partial_\eta \xi_{x_0} + \xi_{y_0}\partial_\xi \xi_{x_0} & \mathbb{L}_{2\mathrm{T}_{3,4}}^{(\mathrm{P_2})} &= \eta_{y_0}\partial_\eta \eta_{x_0} + \xi_{y_0}\partial_\xi \eta_{x_0} \\ \mathbb{L}_{2\mathrm{T}_{2,4}}^{(\mathrm{P_2})} &= \eta_{y_0}\partial_\eta \xi_{y_0} + \xi_{y_0}\partial_\xi \xi_{y_0} & \mathbb{L}_{2\mathrm{T}_{3,4}}^{(\mathrm{P_2})} &= \eta_{y_0}\partial_\eta \eta_{y_0} + \xi_{y_0}\partial_\xi \eta_{y_0} \end{aligned} \tag{3.30}$$

Similarly, the fourth-order mapping is defined as

$$\mathbb{L}_{2\mathrm{T}}^{(\mathrm{P}_3)} \begin{bmatrix} \mathbf{U}_{(0)} \\ \mathbf{U}_{(1)} \\ \mathbf{U}_{(2)} \\ \mathbf{U}_{(3)} \\ \mathbf{U}_{(4)} \\ \mathbf{U}_{(5)} \\ \mathbf{U}_{(6)} \\ \mathbf{U}_{(7)} \\ \mathbf{U}_{(8)} \\ \mathbf{U}_{(9)} \end{bmatrix} = \begin{bmatrix} \mathbf{V}_{(0)} \\ \mathbf{V}_{(1)} \\ \mathbf{V}_{(2)} \\ \mathbf{V}_{(3)} \\ \mathbf{V}_{(4)} \\ \mathbf{V}_{(5)} \\ \mathbf{V}_{(6)} \\ \mathbf{V}_{(7)} \\ \mathbf{V}_{(8)} \\ \mathbf{V}_{(9)} \end{bmatrix} \tag{3.31}$$

$$\mathbb{L}_{2\mathrm{T}}^{(\mathrm{P}_3)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \xi_{x_0} & \eta_{x_0} & 0 & 0 & 0 & \mathbb{L}_{2\mathrm{T}_{7,2}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{8,2}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{9,2}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{10,2}}^{(\mathrm{P}_3)} \\ 0 & \xi_{y_0} & \eta_{y_0} & 0 & 0 & 0 & \mathbb{L}_{2\mathrm{T}_{7,3}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{8,3}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{9,3}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{10,3}}^{(\mathrm{P}_3)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,4}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{3,4}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{4,4}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{5,4}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{6,4}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{7,4}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{8,4}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{9,4}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{10,4}}^{(\mathrm{P}_3)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,5}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{3,5}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{4,5}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{5,5}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{6,5}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{7,5}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{8,5}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{9,5}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{10,5}}^{(\mathrm{P}_3)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,6}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{3,6}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{4,6}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{5,6}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{6,6}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{7,6}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{8,6}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{9,6}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{10,6}}^{(\mathrm{P}_3)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,7}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{3,7}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{4,7}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{5,7}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{6,7}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{7,7}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{8,7}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{9,7}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{10,7}}^{(\mathrm{P}_3)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,8}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{3,8}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{4,8}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{5,8}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{6,8}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{7,8}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{8,8}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{9,8}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{10,8}}^{(\mathrm{P}_3)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,9}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{3,9}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{4,9}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{5,9}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{6,9}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{7,9}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{8,9}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{9,9}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{10,9}}^{(\mathrm{P}_3)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,10}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{3,10}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{4,10}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{5,10}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{6,10}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{7,10}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{8,10}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{9,10}}^{(\mathrm{P}_3)} & \mathbb{L}_{2\mathrm{T}_{10,10}}^{(\mathrm{P}_3)} \end{bmatrix} \tag{3.32}$$

For this and all other mapping matrices, we only show non-zero pattern, for brevity.

The third-order mapping $\mathbb{L}_{2\mathrm{T}}^{(\mathrm{P}_2)}$ in 3D will be defined as

$$\mathbb{L}_{2\mathrm{T}}^{(\mathrm{P}_2)} \begin{bmatrix} \mathbf{U}_{(0)} \\ \mathbf{U}_{(1)} \\ \mathbf{U}_{(2)} \\ \mathbf{U}_{(3)} \\ \mathbf{U}_{(4)} \\ \mathbf{U}_{(5)} \\ \mathbf{U}_{(6)} \\ \mathbf{U}_{(7)} \\ \mathbf{U}_{(8)} \\ \mathbf{U}_{(9)} \end{bmatrix} = \begin{bmatrix} \mathbf{V}_{(0)} \\ \mathbf{V}_{(1)} \\ \mathbf{V}_{(2)} \\ \mathbf{V}_{(3)} \\ \mathbf{V}_{(4)} \\ \mathbf{V}_{(5)} \\ \mathbf{V}_{(6)} \\ \mathbf{V}_{(7)} \\ \mathbf{V}_{(8)} \\ \mathbf{V}_{(9)} \end{bmatrix} \tag{3.33}$$

where

$$\mathbb{L}_{2\mathrm{T}}^{(\mathrm{P}_2)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \xi_{x_0} & \eta_{x_0} & \zeta_{x_0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \xi_{y_0} & \eta_{y_0} & \zeta_{y_0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \xi_{z_0} & \eta_{z_0} & \zeta_{z_0} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,5}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{3,5}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{4,5}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{5,5}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{6,5}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{7,5}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{8,5}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{9,5}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{10,5}}^{(\mathrm{P}_2)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,6}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{3,6}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{4,6}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{5,6}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{6,6}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{7,6}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{8,6}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{9,6}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{10,6}}^{(\mathrm{P}_2)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,7}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{3,7}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{4,7}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{5,7}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{6,7}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{7,7}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{8,7}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{9,7}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{10,7}}^{(\mathrm{P}_2)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,8}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{3,8}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{4,8}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{5,8}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{6,8}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{7,8}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{8,8}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{9,8}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{10,8}}^{(\mathrm{P}_2)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,9}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{3,9}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{4,9}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{5,9}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{6,9}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{7,9}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{8,9}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{9,9}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{10,9}}^{(\mathrm{P}_2)} \\ 0 & \mathbb{L}_{2\mathrm{T}_{2,10}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{3,10}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{4,10}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{5,10}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{6,10}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{7,10}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{8,10}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{9,10}}^{(\mathrm{P}_2)} & \mathbb{L}_{2\mathrm{T}_{10,10}}^{(\mathrm{P}_2)} \end{bmatrix} \tag{3.34}$$

The fourth-order mapping in 3D is

$$\mathbb{L}_{2\mathrm{T}}^{(\mathrm{P}_3)} \begin{bmatrix} \mathbf{U}_{(0)} \\ \mathbf{U}_{(1)} \\ \mathbf{U}_{(2)} \\ \mathbf{U}_{(3)} \\ \mathbf{U}_{(4)} \\ \mathbf{U}_{(5)} \\ \mathbf{U}_{(6)} \\ \mathbf{U}_{(7)} \\ \mathbf{U}_{(8)} \\ \mathbf{U}_{(9)} \\ \mathbf{U}_{(10)} \\ \mathbf{U}_{(11)} \\ \mathbf{U}_{(12)} \\ \mathbf{U}_{(13)} \\ \mathbf{U}_{(14)} \\ \mathbf{U}_{(15)} \\ \mathbf{U}_{(16)} \\ \mathbf{U}_{(17)} \\ \mathbf{U}_{(18)} \\ \mathbf{U}_{(19)} \end{bmatrix} = \begin{bmatrix} \mathbf{V}_{(0)} \\ \mathbf{V}_{(1)} \\ \mathbf{V}_{(2)} \\ \mathbf{V}_{(3)} \\ \mathbf{V}_{(4)} \\ \mathbf{V}_{(5)} \\ \mathbf{V}_{(6)} \\ \mathbf{V}_{(7)} \\ \mathbf{V}_{(8)} \\ \mathbf{V}_{(9)} \\ \mathbf{V}_{(10)} \\ \mathbf{V}_{(11)} \\ \mathbf{V}_{(12)} \\ \mathbf{V}_{(13)} \\ \mathbf{V}_{(14)} \\ \mathbf{V}_{(15)} \\ \mathbf{V}_{(16)} \\ \mathbf{V}_{(17)} \\ \mathbf{V}_{(18)} \\ \mathbf{V}_{(19)} \end{bmatrix} \tag{3.35}$$

where we skipped the details of the matrix $\mathbb{L}_{2\mathrm{T}}^{(\mathrm{P}_3)}$, for brevity.

3.4.4 In-cell reconstruction

The basic idea of the in-cell reconstruction is to build (reconstruct) additional degrees of freedom locally, at each element, thereby increasing the order of accuracy without solving for corresponding PDEs. Consider for example the $\mathrm{RDG}_{\mathrm{P}_2\mathrm{P}_3}$, in two dimensions[3]. We solve for six degrees of freedom per variable (cell-average, 2 slopes and 3 curvatures), which represents the third-order-accurate piecewise-quadratic solution in the element. We want however to enhance the solution locally to the fourth-order, representing the piecewise-cubic solution in an element as

$$\begin{aligned} \mathbf{U}(\xi,\eta) \;=\; & \underbrace{\mathbf{U}_{(0)} + \mathbf{U}_{(1)}\mathcal{B}_{(1)} + \mathbf{U}_{(2)}\mathcal{B}_{(2)} + \mathbf{U}_{(3)}\mathcal{B}_{(3)} + \mathbf{U}_{(4)}\mathcal{B}_{(4)} + \mathbf{U}_{(5)}\mathcal{B}_{(5)}}_{\text{Solved for}} + \\ & \underbrace{+\mathbf{U}_{(6)}\mathcal{B}_{(6)} + \mathbf{U}_{(7)}\mathcal{B}_{(7)} + \mathbf{U}_{(8)}\mathcal{B}_{(8)} + \mathbf{U}_{(9)}\mathcal{B}_{(9)}}_{\text{Reconstructed}} \end{aligned} \tag{3.36}$$

[3]Extension to a generic $\mathrm{RDG}_{\mathrm{P_n P_m}}$ and three dimensions is straightforward.

In this case, four degrees of freedom $\mathbf{U}_{(6,7,8,9)}$ are reconstructed in each element, using the following *least-squares in-cell reconstruction procedure*.

We seek for the missing DoFs, forming the *normal least-squares* problem:

$$\underbrace{\begin{bmatrix}
\mathbb{M}_{11_A} & \mathbb{M}_{12_A} & \mathbb{M}_{13_A} & \mathbb{M}_{14_A} \\
\mathbb{M}_{11_B} & \mathbb{M}_{12_B} & \mathbb{M}_{13_B} & \mathbb{M}_{14_B} \\
\mathbb{M}_{11_C} & \mathbb{M}_{12_C} & \mathbb{M}_{13_C} & \mathbb{M}_{14_C} \\
\mathbb{M}_{11_D} & \mathbb{M}_{12_D} & \mathbb{M}_{13_D} & \mathbb{M}_{14_D} \\
\mathbb{M}_{21_A} & \mathbb{M}_{22_A} & \mathbb{M}_{23_A} & \mathbb{M}_{24_A} \\
\mathbb{M}_{21_B} & \mathbb{M}_{22_B} & \mathbb{M}_{23_B} & \mathbb{M}_{24_B} \\
\mathbb{M}_{21_C} & \mathbb{M}_{22_C} & \mathbb{M}_{23_C} & \mathbb{M}_{24_C} \\
\mathbb{M}_{21_D} & \mathbb{M}_{22_D} & \mathbb{M}_{23_D} & \mathbb{M}_{24_D} \\
\mathbb{M}_{31_A} & \mathbb{M}_{32_A} & \mathbb{M}_{33_A} & \mathbb{M}_{34_A} \\
\mathbb{M}_{31_B} & \mathbb{M}_{32_B} & \mathbb{M}_{33_B} & \mathbb{M}_{34_B} \\
\mathbb{M}_{31_C} & \mathbb{M}_{32_C} & \mathbb{M}_{33_C} & \mathbb{M}_{34_C} \\
\mathbb{M}_{31_D} & \mathbb{M}_{32_D} & \mathbb{M}_{33_D} & \mathbb{M}_{34_D} \\
\mathbb{M}_{41_A} & \mathbb{M}_{42_A} & \mathbb{M}_{43_A} & \mathbb{M}_{44_A} \\
\mathbb{M}_{41_B} & \mathbb{M}_{42_B} & \mathbb{M}_{43_B} & \mathbb{M}_{44_B} \\
\mathbb{M}_{41_C} & \mathbb{M}_{42_C} & \mathbb{M}_{43_C} & \mathbb{M}_{44_C} \\
\mathbb{M}_{41_D} & \mathbb{M}_{42_D} & \mathbb{M}_{43_D} & \mathbb{M}_{44_D} \\
\mathbb{M}_{51_A} & \mathbb{M}_{52_A} & \mathbb{M}_{53_A} & \mathbb{M}_{54_A} \\
\mathbb{M}_{51_B} & \mathbb{M}_{52_B} & \mathbb{M}_{53_B} & \mathbb{M}_{54_B} \\
\mathbb{M}_{51_C} & \mathbb{M}_{52_C} & \mathbb{M}_{53_C} & \mathbb{M}_{54_C} \\
\mathbb{M}_{51_D} & \mathbb{M}_{52_D} & \mathbb{M}_{53_D} & \mathbb{M}_{54_D} \\
\mathbb{M}_{61_A} & \mathbb{M}_{62_A} & \mathbb{M}_{63_A} & \mathbb{M}_{64_A} \\
\mathbb{M}_{61_B} & \mathbb{M}_{62_B} & \mathbb{M}_{63_B} & \mathbb{M}_{64_B} \\
\mathbb{M}_{61_C} & \mathbb{M}_{62_C} & \mathbb{M}_{63_C} & \mathbb{M}_{64_C} \\
\mathbb{M}_{61_D} & \mathbb{M}_{62_D} & \mathbb{M}_{63_D} & \mathbb{M}_{64_D}
\end{bmatrix}}_{\mathbb{M}}
\begin{bmatrix}
\mathbf{U}_{(6)} \\ \mathbf{U}_{(7)} \\ \mathbf{U}_{(8)} \\ \mathbf{U}_{(9)}
\end{bmatrix} =
\underbrace{\begin{bmatrix}
\mathcal{R}_{1_A} \\ \mathcal{R}_{1_B} \\ \mathcal{R}_{1_C} \\ \mathcal{R}_{1_D} \\
\mathcal{R}_{2_A} \\ \mathcal{R}_{2_B} \\ \mathcal{R}_{2_C} \\ \mathcal{R}_{2_D} \\
\mathcal{R}_{3_A} \\ \mathcal{R}_{3_B} \\ \mathcal{R}_{3_C} \\ \mathcal{R}_{3_D} \\
\mathcal{R}_{4_A} \\ \mathcal{R}_{4_B} \\ \mathcal{R}_{4_C} \\ \mathcal{R}_{4_D} \\
\mathcal{R}_{5_A} \\ \mathcal{R}_{5_B} \\ \mathcal{R}_{5_C} \\ \mathcal{R}_{5_D} \\
\mathcal{R}_{6_A} \\ \mathcal{R}_{6_B} \\ \mathcal{R}_{6_C} \\ \mathcal{R}_{6_D}
\end{bmatrix}}_{\mathcal{R}} \tag{3.37}$$

Each of these 24 equations represents a constraint, being enforced on the reconstructed solution polinomial, in the least squares sense.

Cell-centered constraints

The first four equations in the system (3.37) are formulated by requiring the in-cell solution eq.(3.36) to reconstruct the *cell-centered (point-wise) solutions* in four[4] face-neighboring cells A, B, C and D, when extended beyond the cell E under the consideration. The "extended" solution profile can be written as

$$\mathbf{U}\left(\bar{x},\bar{y}\right)=\mathbb{L}_{2\mathrm{T_E}}^{(\mathrm{P_3})}\left[\begin{array}{c}\mathbf{U}_{(0)}\\ \mathbf{U}_{(1)}\\ \mathbf{U}_{(2)}\\ \mathbf{U}_{(3)}\\ \mathbf{U}_{(4)}\\ \mathbf{U}_{(5)}\\ \mathbf{U}_{(6)}\\ \mathbf{U}_{(7)}\\ \mathbf{U}_{(8)}\\ \mathbf{U}_{(9)}\end{array}\right]_{\mathrm{E}}\left[\begin{array}{c}\mathcal{T}_{(0)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(1)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(2)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(3)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(4)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(5)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(6)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(7)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(8)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(9)}\left(\bar{x},\bar{y}\right)\end{array}\right] \tag{3.38}$$

where $\mathcal{T}_{(n)}$ are the Taylor basis functions, as defined by eq.(3.6), while $\bar{x}=x-x_{\mathrm{E}}$ and $\bar{y}=y-y_{\mathrm{E}}$. Thus, each of the first four equations in (3.37) can be formed by setting $(\bar{x},\bar{y})$ to $(\bar{x}_{\mathrm{A}},\bar{y}_{\mathrm{A}})$, $(\bar{x}_{\mathrm{B}},\bar{y}_{\mathrm{B}})$, $(\bar{x}_{\mathrm{C}},\bar{y}_{\mathrm{C}})$ and $(\bar{x}_{\mathrm{D}},\bar{y}_{\mathrm{D}})$, and then collecting the coefficients in the front of $\mathbf{U}_{(6,7,8,9)}$ – these are the $\mathbb{M}_{1\alpha_\beta}$ in eq.(3.37), while the rest terms are stashed into the right-hand-side terms $\mathcal{R}_{1_\beta}$, where $\alpha=1,2,3,4$ and $\beta=$A,B,C and D.

Notably, the cell-centered solutions in the neighbor cells are directly available from the DoFs solved for, to the order of interest:

$$\begin{aligned}\mathbf{U}\left(x_\beta,y_\beta\right)=&\mathbf{U}_{(0)_\beta}-\\&-\tfrac{1}{\Omega_\beta}\int\limits_{\Omega_\beta}\left(\mathbf{V}_{(3)_\beta}\tfrac{\left(x-x_\beta\right)^2}{2}+\mathbf{V}_{(4)_\beta}\left(x-x_\beta\right)\left(y-y_\beta\right)+\mathbf{V}_{(5)_\beta}\tfrac{\left(y-y_\beta\right)^2}{2}\right)d\Omega\end{aligned} \tag{3.39}$$

where $\mathbf{V}_{(3,4,5)_\beta}$ are computed using eq.(3.29).

This assemble procedure can be easily coded for a generic set of the $\mathrm{RDG}_{\mathrm{P_nP_m}}$ reconstruction/mesh geometry.

[4] In the discussion, we are considering QUAD elements. Extension to HEX elements is straightforward.

Slope-consistency constraints

The next eight equations represent the requirement for the "extended" solution shape to be consistent with the *point-wise solution derivatives*, in the face-neighboring cells A, B, C and D. For derivatives in x-directions,

$$\frac{\partial}{\partial \bar{x}}\mathbf{U}\left(\bar{x},\bar{y}\right)=\mathbb{L}_{2\mathrm{T_E}}^{(\mathrm{P}_3)}\begin{bmatrix}\mathbf{U}_{(0)}\\ \mathbf{U}_{(1)}\\ \mathbf{U}_{(2)}\\ \mathbf{U}_{(3)}\\ \mathbf{U}_{(4)}\\ \mathbf{U}_{(5)}\\ \mathbf{U}_{(6)}\\ \mathbf{U}_{(7)}\\ \mathbf{U}_{(8)}\\ \mathbf{U}_{(9)}\end{bmatrix}_{\mathrm{E}}\frac{\partial}{\partial \bar{x}}\begin{bmatrix}\mathcal{T}_{(0)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(1)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(2)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(3)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(4)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(5)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(6)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(7)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(8)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(9)}\left(\bar{x},\bar{y}\right)\end{bmatrix} \tag{3.40}$$

The assemble procedure is straightforward – set $(\bar{x},\bar{y})$ to $(\bar{x}_{\mathrm{A}},\bar{y}_{\mathrm{A}})$, $(\bar{x}_{\mathrm{B}},\bar{y}_{\mathrm{B}})$, $(\bar{x}_{\mathrm{C}},\bar{y}_{\mathrm{C}})$ and $(\bar{x}_{\mathrm{D}},\bar{y}_{\mathrm{D}})$, collect the coefficients in the front of $\mathbf{U}_{(6,7,8,9)}$ – these are the $\mathbb{M}_{2\alpha_\beta}$ in eq.(3.37), and collect all the rest into right-hand-side terms $\mathcal{R}_{2_\beta}$.

The cell-centered derivatives $\mathbf{V}_{(1)_\beta}$ and $\mathbf{V}_{(2)_\beta}$ in the neighbor cells are available from the DoFs solved for, to the order of interest (the 4^{th}), using eq.(3.29).

Similar procedure is applied for the slopes in y-direction, to assemble the $\mathbb{M}_{3\alpha_\beta}$ and $\mathcal{R}_{3_\beta}$, using

$$\frac{\partial}{\partial \bar{y}}\mathbf{U}\left(\bar{x},\bar{y}\right)=\mathbb{L}_{2\mathrm{T_E}}^{(\mathrm{P}_3)}\begin{bmatrix}\mathbf{U}_{(0)}\\ \mathbf{U}_{(1)}\\ \mathbf{U}_{(2)}\\ \mathbf{U}_{(3)}\\ \mathbf{U}_{(4)}\\ \mathbf{U}_{(5)}\\ \mathbf{U}_{(6)}\\ \mathbf{U}_{(7)}\\ \mathbf{U}_{(8)}\\ \mathbf{U}_{(9)}\end{bmatrix}_{\mathrm{E}}\frac{\partial}{\partial \bar{y}}\begin{bmatrix}\mathcal{T}_{(0)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(1)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(2)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(3)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(4)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(5)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(6)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(7)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(8)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(9)}\left(\bar{x},\bar{y}\right)\end{bmatrix} \tag{3.41}$$

Curvature-consistency constraints

All the rest 12 equations are designed to enforce the *point-wise curvature consistency*. For the second derivatives in x-direction,

$$\frac{\partial^2}{\partial \bar{x}^2}\mathbf{U}\left(\bar{x},\bar{y}\right)=\mathbb{L}_{_{2\mathrm{T_E}}}^{(\mathrm{P}_3)}\left[\begin{array}{c}\mathbf{U}_{(0)}\\ \mathbf{U}_{(1)}\\ \mathbf{U}_{(2)}\\ \mathbf{U}_{(3)}\\ \mathbf{U}_{(4)}\\ \mathbf{U}_{(5)}\\ \mathbf{U}_{(6)}\\ \mathbf{U}_{(7)}\\ \mathbf{U}_{(8)}\\ \mathbf{U}_{(9)}\end{array}\right]_{\mathrm{E}}\frac{\partial^2}{\partial \bar{x}^2}\left[\begin{array}{c}\mathcal{T}_{(0)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(1)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(2)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(3)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(4)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(5)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(6)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(7)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(8)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(9)}\left(\bar{x},\bar{y}\right)\end{array}\right] \tag{3.42}$$

which can be used to assemble the $\mathbb{M}_{_{4\alpha_\beta}}$ and $\mathcal{R}_{_{4_\beta}}$.

To get the $\mathbb{M}_{_{5\alpha_\beta}}$ and $\mathcal{R}_{_{5_\beta}}$, one can use

$$\frac{\partial^2}{\partial \bar{x}\,\partial \bar{y}}\mathbf{U}\left(\bar{x},\bar{y}\right)=\mathbb{L}_{_{2\mathrm{T_E}}}^{(\mathrm{P}_3)}\left[\begin{array}{c}\mathbf{U}_{(0)}\\ \mathbf{U}_{(1)}\\ \mathbf{U}_{(2)}\\ \mathbf{U}_{(3)}\\ \mathbf{U}_{(4)}\\ \mathbf{U}_{(5)}\\ \mathbf{U}_{(6)}\\ \mathbf{U}_{(7)}\\ \mathbf{U}_{(8)}\\ \mathbf{U}_{(9)}\end{array}\right]_{\mathrm{E}}\frac{\partial^2}{\partial \bar{x}\,\partial \bar{y}}\left[\begin{array}{c}\mathcal{T}_{(0)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(1)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(2)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(3)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(4)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(5)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(6)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(7)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(8)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(9)}\left(\bar{x},\bar{y}\right)\end{array}\right] \tag{3.43}$$

while

$$\frac{\partial^2}{\partial \bar{y}^2}\mathbf{U}\left(\bar{x},\bar{y}\right)=\mathbb{L}_{_{2\mathrm{T_E}}}^{(\mathrm{P}_3)}\left[\begin{array}{c}\mathbf{U}_{(0)}\\ \mathbf{U}_{(1)}\\ \mathbf{U}_{(2)}\\ \mathbf{U}_{(3)}\\ \mathbf{U}_{(4)}\\ \mathbf{U}_{(5)}\\ \mathbf{U}_{(6)}\\ \mathbf{U}_{(7)}\\ \mathbf{U}_{(8)}\\ \mathbf{U}_{(9)}\end{array}\right]_{\mathrm{E}}\frac{\partial^2}{\partial \bar{y}^2}\left[\begin{array}{c}\mathcal{T}_{(0)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(1)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(2)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(3)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(4)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(5)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(6)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(7)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(8)}\left(\bar{x},\bar{y}\right)\\ \mathcal{T}_{(9)}\left(\bar{x},\bar{y}\right)\end{array}\right] \tag{3.44}$$

is utilized to assemble the $\mathbb{M}_{6\alpha_\beta}$ and $\mathcal{R}_{6_\beta}$.

The cell-centered curvatures $\mathbf{V}_{(3)_\beta}$, $\mathbf{V}_{(4)_\beta}$ and $\mathbf{V}_{(5)_\beta}$ in the neighbor cells are available from the DoFs solved for, to the 4^{th} order of interest, using eq.(3.29).

Least-Squares Solution

Once the non-square matrix $\mathbb{M}$ and the *r.h.s.* vector $\boldsymbol{\mathcal{R}}$ are assembled, the least-squares problem eq.(3.37) can be solved for $\mathbf{U}_{(6,7,8,9)}$ as:

$$\begin{bmatrix} \mathbf{U}_{(6)} \\ \mathbf{U}_{(7)} \\ \mathbf{U}_{(8)} \\ \mathbf{U}_{(9)} \end{bmatrix} = \left(\mathbb{M}^{\mathrm{T}}\mathbb{M}\right)^{-1}\left(\mathbb{M}^{\mathrm{T}}\boldsymbol{\mathcal{R}}\right) \tag{3.45}$$

Importantly, before applying eq.(3.45), both the $\mathbb{M}$ and $\boldsymbol{\mathcal{R}}$ are normalized by first finding the element of the matrix $\mathbb{M}$ with the largest absolute value, $\mathbb{M}_{\max}$, and then dividing both the $\mathbb{M}$ and $\boldsymbol{\mathcal{R}}$ on $\mathbb{M}_{\max}$. This helps to get a better conditioned matrix inversion in eq.(3.45), which is especially relevant for the higher-order ($> 3^{\text{rd}}$) reconstructions.

Blended recovery/reconstruction operator

We found that a better (more accurate) solution can be achieved by adding integral (weak) statements into the least squares formulation eq.(3.37). In this case, the least-squares problem become:

$$\underbrace{\left[\begin{array}{cccc} \mathbb{M}_{11_{\mathrm{A}}} & \mathbb{M}_{12_{\mathrm{A}}} & \mathbb{M}_{13_{\mathrm{A}}} & \mathbb{M}_{14_{\mathrm{A}}} \\ & \cdots & & \\ \mathbb{M}_{61_{\mathrm{D}}} & \mathbb{M}_{62_{\mathrm{D}}} & \mathbb{M}_{63_{\mathrm{D}}} & \mathbb{M}_{64_{\mathrm{D}}} \\ \hline \mathbb{M}_{71_{\mathrm{o}}} & \mathbb{M}_{72_{\mathrm{o}}} & \mathbb{M}_{73_{\mathrm{o}}} & \mathbb{M}_{74_{\mathrm{o}}} \\ & \cdots & & \end{array}\right]}_{\mathbb{M}} \begin{bmatrix} \mathbf{U}_{(6)} \\ \mathbf{U}_{(7)} \\ \mathbf{U}_{(8)} \\ \mathbf{U}_{(9)} \end{bmatrix} = \underbrace{\left[\begin{array}{c} \mathcal{R}_{1_{\mathrm{A}}} \\ \cdots \\ \mathcal{R}_{6_{\mathrm{D}}} \\ \hline \\ \mathcal{R}_{7_{\mathrm{o}}} \\ \cdots \end{array}\right]}_{\boldsymbol{\mathcal{R}}} \tag{3.46}$$

where the equations

$$\mathbb{M}_{71_{\mathrm{o}}}\mathbf{U}_{(6)} + \mathbb{M}_{72_{\mathrm{o}}}\mathbf{U}_{(7)} + \mathbb{M}_{73_{\mathrm{o}}}\mathbf{U}_{(8)} + \mathbb{M}_{74_{\mathrm{o}}}\mathbf{U}_{(9)} = \mathcal{R}_{7_{\mathrm{o}}}, \quad \mathrm{o} = 0, ..., \mathrm{N}_7 \tag{3.47}$$

are added to enforce (in the least-squares sense) integral (recovery) properties. In eq.(3.47), N_{7} could optionally be 0, 2 or 5, depending on the order of weak statements decided to be "blended in" (cell-average-, slope- or curvature- "integral consistency"). In most cases considered here, we enforce only the cell-average ($\mathrm{N}_{7} = 0$) consistency.

More specifically, the integral statements being implemented are

$$\int_{\Omega_\beta} \omega_{_w} \mathbf{U}\left(x,y\right) \mathcal{B}_{_{(\mathrm{o})_\beta}}\left(x,y\right) d\Omega = \frac{\omega_{_w} \mathbf{U}_{_{(\mathrm{o})_\beta}}}{\mathcal{A}_{_{(\mathrm{o})}}} \tag{3.48}$$

where the $\omega_{_w}$ is the weight of the integral statements (in most cases considered here – set to 10, i.e. contributing ten times more than the strong statements); the $\mathbf{U}\left(x,y\right)$ is the reconstructed solution in the cell under consideration, eq.(3.36); the $\mathcal{B}_{_{(\mathrm{o})_\beta}}$ is the orthogonal basis function (the o^{th}-order) in the neighbor cell β; the $\mathcal{A}_{_{(\mathrm{o})}}$ is the normalization coefficient, as defined by eq.(3.26); and the $\mathbf{U}_{_{(\mathrm{o})_\beta}}$ is the o^{th}-order DoF in the neighbor cell β. The eq.(3.48) corresponds to the constraints utilized in the *in-cell recovery schemes*, as introduced in [NTP+08, NPM09]. This enforces the Sobolev-space consistency of the reconstructed fields, with the "solved for" DoFs – here, in the least-squares sense.

To convert eq.(3.48) to the discrete form eq.(3.47), we replace the integral with the summation over the Gaussian integration points in the neighbor cells β, collect the coefficients in the front of $\mathbf{U}_{_{(6,7,8,9)}}$ – these are the $\mathbb{M}_{_{7\alpha_{\mathrm{o}}}}$, and stash all the rest terms into the *r.h.s.* terms $\mathcal{R}_{_{7_{\mathrm{o}}}}$. This assembling procedure can be easily generalized to any order and in three dimensions.

Boundary conditions

Boundary elements require special consideration. In this case, some of the neighbor elements are missing. While it is totally acceptable to just drop corresponding equations from the least-squares formulation eq.(3.37) – the resulting linear algebra eq.(3.45) is still well-behaved, we do enforce boundary conditions in the in-cell reconstruction, as discussed below. We believe that infusing BC into the reconstruction results in a better conditioned and more accurate discretization scheme.

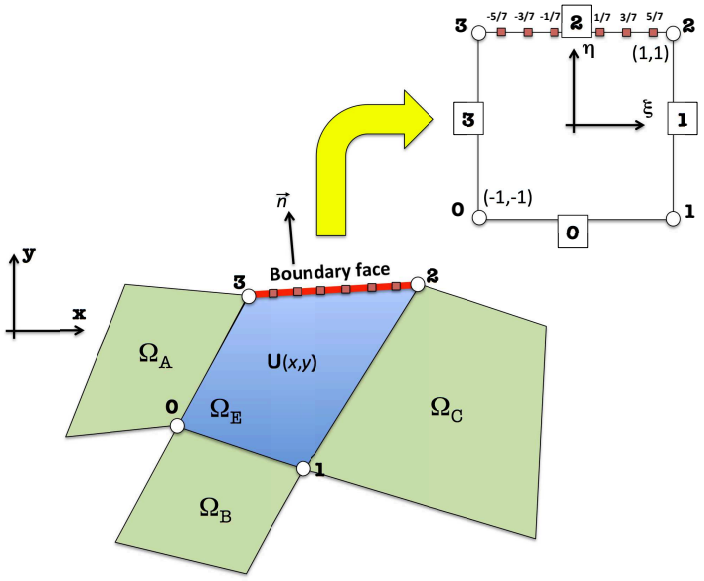


Fig. 3.3 : On treatment of boundary conditions for the $\mathrm{RDG}_{\mathrm{P}_2\mathrm{P}_3}$ in-cell reconstruction.

Without loss of generality, consider a boundary element as depicted in Figure 3.3. In the considered case, the face at $\eta = 1$ is at the boundary. Suppose we want to enforce **Dirichlet BC**[5]. On the boundary face – we define six boundary nodes, with the reference-space coordinates $\boldsymbol{\chi}_{n=0,..,5} = \left(-\frac{1}{7}, 1\right), \left(-\frac{3}{7}, 1\right), \left(-\frac{5}{7}, 1\right), \left(\frac{1}{7}, 1\right), \left(\frac{3}{7}, 1\right)$ and $\left(\frac{5}{7}, 1\right)$[6]. The boundary condition values at these nodes are as-

[5]Extension to other types of boundary conditions is straightforward.

[6]For other orders and element types, for the face with $\eta = 1$, we use the following face boundary node reference-space coordinates:

Element type	Reconstruction	$\boldsymbol{\chi}_n$
QUAD	$\mathrm{RDG}_{\mathrm{P}_0\mathrm{P}_1}$ $\mathrm{RDG}_{\mathrm{P}_1\mathrm{P}_2}$ $\mathrm{RDG}_{\mathrm{P}_2\mathrm{P}_3}$	$(0,1)$ $\left(\pm\frac{1}{2}, 1\right), \quad (0,1)$ $\left(\pm\frac{1}{7}, 1\right), \quad \left(\pm\frac{3}{7}, 1\right), \quad \left(\pm\frac{5}{7}, 1\right)$
HEX	$\mathrm{RDG}_{\mathrm{P}_0\mathrm{P}_1}$ $\mathrm{RDG}_{\mathrm{P}_1\mathrm{P}_2}$	$(0,1,0)$ $\left(\pm\frac{1}{2}, 1, 0\right), \quad \left(0, 1, \pm\frac{1}{2}\right), \quad (0,1,0)$

(3.49)

sumed to be available, generally as a function of time, $\mathcal{F}_n(t)$. To enforce these Dirichlet BC values, we replace the "missing neighbor" sub-block in eq.(3.37), with the following sub-block

$$\omega_{\text{BC}} \begin{bmatrix} \frac{55}{343} & \frac{13}{49} & -\frac{5}{7} & 1 \\ \frac{153}{343} & -\frac{11}{49} & -\frac{3}{7} & 1 \\ \frac{71}{343} & -\frac{23}{49} & -\frac{1}{7} & 1 \\ -\frac{71}{343} & -\frac{23}{49} & \frac{1}{7} & 1 \\ -\frac{153}{343} & -\frac{11}{49} & \frac{3}{7} & 1 \\ -\frac{55}{343} & \frac{13}{49} & \frac{5}{7} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{U}_{(6)} \\ \mathbf{U}_{(7)} \\ \mathbf{U}_{(8)} \\ \mathbf{U}_{(9)} \end{bmatrix} = \omega_{\text{BC}} \begin{bmatrix} \mathcal{F}_0 - \left(\mathbf{U}_{(0)} - \frac{5}{7}\mathbf{U}_{(1)} + \mathbf{U}_{(2)} + \frac{13}{49}\mathbf{U}_{(3)} - \frac{5}{7}\mathbf{U}_{(4)} + \mathbf{U}_{(5)}\right) \\ \mathcal{F}_1 - \left(\mathbf{U}_{(0)} - \frac{3}{7}\mathbf{U}_{(1)} + \mathbf{U}_{(2)} - \frac{11}{49}\mathbf{U}_{(3)} - \frac{3}{7}\mathbf{U}_{(4)} + \mathbf{U}_{(5)}\right) \\ \mathcal{F}_2 - \left(\mathbf{U}_{(0)} - \frac{1}{7}\mathbf{U}_{(1)} + \mathbf{U}_{(2)} - \frac{23}{49}\mathbf{U}_{(3)} - \frac{1}{7}\mathbf{U}_{(4)} + \mathbf{U}_{(5)}\right) \\ \mathcal{F}_3 - \left(\mathbf{U}_{(0)} + \frac{1}{7}\mathbf{U}_{(1)} + \mathbf{U}_{(2)} - \frac{23}{49}\mathbf{U}_{(3)} + \frac{1}{7}\mathbf{U}_{(4)} + \mathbf{U}_{(5)}\right) \\ \mathcal{F}_4 - \left(\mathbf{U}_{(0)} + \frac{3}{7}\mathbf{U}_{(1)} + \mathbf{U}_{(2)} - \frac{11}{49}\mathbf{U}_{(3)} + \frac{3}{7}\mathbf{U}_{(4)} + \mathbf{U}_{(5)}\right) \\ \mathcal{F}_5 - \left(\mathbf{U}_{(0)} + \frac{5}{7}\mathbf{U}_{(1)} + \mathbf{U}_{(2)} + \frac{13}{49}\mathbf{U}_{(3)} + \frac{5}{7}\mathbf{U}_{(4)} + \mathbf{U}_{(5)}\right) \end{bmatrix} \tag{3.50}$$

These six equations are derived to enforce the reconstructed in-cell polynomial eq.(3.36) to be equal to the boundary values $\mathcal{F}_n$. This will be satisfied in the least-squares sense, and how close – will be controlled by the variable BC weight, ω_{BC}, usually set in the range from 1 to 10.

The **Neumann boundary conditions** are enforced similarly. Given the normal derivatives of the function at boundary nodes $\left.\frac{d\mathcal{F}}{d\vec{n}}\right|_k(t)$, the "replacement" sub-block is

$$\omega_{\text{BC}}\mathbb{M}_{\text{Neumann}} \begin{bmatrix} \mathbf{U}_{(6)} \\ \mathbf{U}_{(7)} \\ \mathbf{U}_{(8)} \\ \mathbf{U}_{(9)} \end{bmatrix} = -\omega_{\text{BC}} \begin{bmatrix} \left.\frac{d\mathcal{F}}{dn}\right|_0 + \frac{\mathcal{D}_0}{7} \\ \left.\frac{d\mathcal{F}}{dn}\right|_1 + \frac{\mathcal{D}_1}{7} \\ \left.\frac{d\mathcal{F}}{dn}\right|_2 + \frac{\mathcal{D}_2}{7} \\ \left.\frac{d\mathcal{F}}{dn}\right|_3 + \frac{\mathcal{D}_3}{7} \\ \left.\frac{d\mathcal{F}}{dn}\right|_4 + \frac{\mathcal{D}_4}{7} \\ \left.\frac{d\mathcal{F}}{dn}\right|_5 + \frac{\mathcal{D}_5}{7} \end{bmatrix} \tag{3.51}$$

where $\vec{n} = \left(n_x, n_y\right)$ is the normal-to-the-boundary vector. The matrix $\mathbb{M}_{\text{Neumann}}$ is

Similar for the faces with $\xi = \pm 1$, $\eta = -1$ and $\zeta = \pm 1$.

defined as

$$\mathbb{M}_{\text{Neumann}} = \begin{bmatrix} -\frac{114}{49}a_0 & \frac{1}{49}\left(105a_0 - 13b_0\right) & -a_0 + \frac{15}{7}b_0 & -6b_0 \\ \frac{6}{49}a_1 & \frac{1}{49}\left(63a_1 + 11b_1\right) & -a_1 + \frac{9}{7}b_1 & -6b_1 \\ \frac{66}{49}a_2 & \frac{1}{49}\left(21a_2 + 23b_2\right) & -a_2 + \frac{3}{7}b_2 & -6b_2 \\ \frac{66}{49}a_3 & \frac{1}{49}\left(-21a_3 + 23b_3\right) & -a_3 - \frac{3}{7}b_3 & -6b_3 \\ \frac{6}{49}a_4 & \frac{1}{49}\left(-63a_4 + 11b_4\right) & -a_4 - \frac{9}{7}b_4 & -6b_4 \\ -\frac{114}{49}a_5 & \frac{1}{49}\left(-105a_5 - 13b_5\right) & -a_5 - \frac{15}{7}b_5 & -6b_5 \end{bmatrix} \tag{3.52}$$

where

$$\begin{aligned} a_k &= \xi_{x_{(k)}} n_x + \xi_{y_{(k)}} n_y \\ b_k &= \eta_{x_{(k)}} n_x + \eta_{y_{(k)}} n_y \end{aligned} \tag{3.53}$$

and $\xi_{x_{(k)}}$, $\xi_{y_{(k)}}$, $\eta_{x_{(k)}}$ and $\eta_{y_{(k)}}$ are components of the inverse of the Jacobian eq.(3.14), evaluated at the k^{th} boundary node. The *r.h.s.* terms $\mathcal{D}_k$ are defined as

$$\begin{aligned} \mathcal{D}_0 &= a_0\left(-7\mathbf{U}_{(1)} + 15\mathbf{U}_{(3)} - 7\mathbf{U}_{(4)}\right) + b_0\left(-7\mathbf{U}_{(2)} + 5\mathbf{U}_{(4)} - 21\mathbf{U}_{(5)}\right) \\ \mathcal{D}_1 &= a_1\left(-7\mathbf{U}_{(1)} + 9\mathbf{U}_{(3)} - 7\mathbf{U}_{(4)}\right) + b_1\left(-7\mathbf{U}_{(2)} + 3\mathbf{U}_{(4)} - 21\mathbf{U}_{(5)}\right) \\ \mathcal{D}_2 &= a_2\left(-7\mathbf{U}_{(1)} + 3\mathbf{U}_{(3)} - 7\mathbf{U}_{(4)}\right) + b_2\left(-7\mathbf{U}_{(2)} + \mathbf{U}_{(4)} - 21\mathbf{U}_{(5)}\right) \\ \mathcal{D}_3 &= -a_3\left(7\mathbf{U}_{(1)} + 3\mathbf{U}_{(3)} + 7\mathbf{U}_{(4)}\right) - b_3\left(7\mathbf{U}_{(2)} + \mathbf{U}_{(4)} + 21\mathbf{U}_{(5)}\right) \\ \mathcal{D}_4 &= -a_4\left(7\mathbf{U}_{(1)} + 9\mathbf{U}_{(3)} + 7\mathbf{U}_{(4)}\right) - b_4\left(7\mathbf{U}_{(2)} + 3\mathbf{U}_{(4)} + 21\mathbf{U}_{(5)}\right) \\ \mathcal{D}_5 &= -a_5\left(7\mathbf{U}_{(1)} + 15\mathbf{U}_{(3)} + 7\mathbf{U}_{(4)}\right) - b_5\left(7\mathbf{U}_{(2)} + 5\mathbf{U}_{(4)} + 21\mathbf{U}_{(5)}\right) \end{aligned} \tag{3.54}$$

The six equations (3.51) are derived to enforce the normal derivatives of the reconstructed in-cell polynomial eq.(3.36) to be equal to the given boundary normal derivatives $\frac{d\mathcal{F}}{d\vec{n}}\Big|_k$. Similarly to the Dirichlet BC, this will be satisfied in the least-squares sense, with the "dial" ω_{BC}, controlling the weight of these constraints after eq.(3.45) is applied.

Generalization

It is understood that different-order reconstructions are implemented in the similar fashion. For example, in 2D $\text{RDG}_{\text{P}_0\text{P}_1}$, we solve for the cell averages (finite-

volume formulation), and reconstruct the slopes:

$$\mathbf{U}(\xi,\eta) = \underbrace{\mathbf{U}_{(0)}}_{\text{Solved for}} + \underbrace{\mathbf{U}_{(1)}\mathcal{B}_{(1)} + \mathbf{U}_{(2)}\mathcal{B}_{(2)}}_{\text{Reconstructed}} \tag{3.55}$$

using the cell-centered constraints. Similarly, for 2D $\mathrm{RDG}_{\mathrm{P_1P_2}}$, we solve for the cell averages and slopes, reconstructing the curvatures

$$\mathbf{U}(\xi,\eta) = \underbrace{\mathbf{U}_{(0)} + \mathbf{U}_{(1)}\mathcal{B}_{(1)} + \mathbf{U}_{(2)}\mathcal{B}_{(2)}}_{\text{Solved for}} + \underbrace{\mathbf{U}_{(3)}\mathcal{B}_{(3)} + \mathbf{U}_{(4)}\mathcal{B}_{(4)} + \mathbf{U}_{(5)}\mathcal{B}_{(5)}}_{\text{Reconstructed}} \tag{3.56}$$

utilizing the cell-centered and slope consistency constraints.

It is also feasible to developed a more "aggressive" reconstruction[7], for example the $\mathrm{RDG}_{\mathrm{P_1P_3}}$ in 2D:

$$\begin{aligned}\mathbf{U}(\xi,\eta) = & \underbrace{\mathbf{U}_{(0)} + \mathbf{U}_{(1)}\mathcal{B}_{(1)} + \mathbf{U}_{(2)}\mathcal{B}_{(2)}}_{\text{Solved for}} + \\ & + \underbrace{\mathbf{U}_{(3)}\mathcal{B}_{(3)} + \mathbf{U}_{(4)}\mathcal{B}_{(4)} + \mathbf{U}_{(5)}\mathcal{B}_{(5)} + \mathbf{U}_{(6)}\mathcal{B}_{(6)} + \mathbf{U}_{(7)}\mathcal{B}_{(7)} + \mathbf{U}_{(8)}\mathcal{B}_{(8)} + \mathbf{U}_{(9)}\mathcal{B}_{(9)}}_{\text{Reconstructed}}\end{aligned} \tag{3.57}$$

3.4.5 Inter-cell reconstruction

The inter-cell reconstruction is applied to compute parabolic (and elliptic) operators. For these, we consider the unions of the face-neighboring cells. For each pair, we reconstruct a "consistent-with-parabolic-operator" solution profile, which is used for compute diffusion flux at each Gaussian point of the corresponding face.

Without loss of generality, we will discuss a union of cells E and A, as depicted in Figure 3.4. For 2D $\mathrm{RDG}_{\mathrm{P_2P_3}}$, the *inter-cell reconstructed solution* is

$$\begin{aligned}\mathbf{U}(\bar{x},\bar{y}) = \mathbf{V}_{(0)} & + \mathbf{V}_{(1)}\mathcal{T}_{(1)}(\bar{x},\bar{y}) + \mathbf{V}_{(2)}\mathcal{T}_{(2)}(\bar{x},\bar{y}) + \mathbf{V}_{(3)}\mathcal{T}_{(3)}(\bar{x},\bar{y}) + \\ & + \mathbf{V}_{(4)}\mathcal{T}_{(4)}(\bar{x},\bar{y}) + \mathbf{V}_{(5)}\mathcal{T}_{(5)}(\bar{x},\bar{y}) + \mathbf{V}_{(6)}\mathcal{T}_{(6)}(\bar{x},\bar{y}) + \\ & + \mathbf{V}_{(7)}\mathcal{T}_{(7)}(\bar{x},\bar{y}) + \mathbf{V}_{(8)}\mathcal{T}_{(8)}(\bar{x},\bar{y}) + \mathbf{V}_{(9)}\mathcal{T}_{(9)}(\bar{x},\bar{y})\end{aligned} \tag{3.58}$$

where $\bar{x} = x - x_{\mathrm{AE}}$, $\bar{y} = y - y_{\mathrm{AE}}$, and $(x_{\mathrm{AE}}, y_{\mathrm{AE}}) = \left(\frac{x_{\mathrm{A}}+x_{\mathrm{E}}}{2}, \frac{y_{\mathrm{A}}+y_{\mathrm{E}}}{2}\right)$. The degrees of freedom being reconstructed are $\mathbf{V}_{(0,\ldots,9)}$. Note that these are based on the

[7] Though, not attempted here.

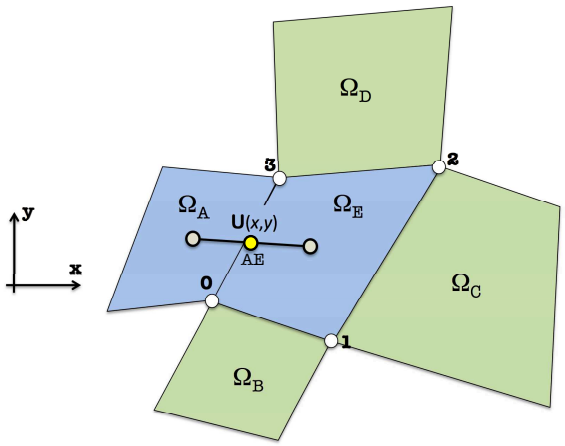


Fig. 3.4 : On intercell reconstruction.

Taylor basis functions, as it is more consistent/convenient with the intended use in diffusion operators (the Fourier fluxes and the Navier-Stokes stress tensor).

To find these DoFs, we form the following *normal least-squares* problem:

$$\underbrace{\left[\begin{array}{cccc} \mathbb{M}_{0,0_A} & \mathbb{M}_{0,1_A} & \dots & \mathbb{M}_{0,9_A} \\ \mathbb{M}_{0,0_E} & \mathbb{M}_{0,1_E} & \dots & \mathbb{M}_{0,9_E} \\ & & & \\ \mathbb{M}_{1,0_A} & \mathbb{M}_{1,1_A} & \dots & \mathbb{M}_{1,9_A} \\ \mathbb{M}_{1,0_E} & \mathbb{M}_{1,1_E} & \dots & \mathbb{M}_{1,9_E} \\ & \dots & & \\ \mathbb{M}_{9,0_A} & \mathbb{M}_{9,1_A} & \dots & \mathbb{M}_{9,9_A} \\ \mathbb{M}_{9,0_E} & \mathbb{M}_{9,1_E} & \dots & \mathbb{M}_{9,9_E} \\ \hline & & & \\ \mathbb{M}_{10,0_o} & \mathbb{M}_{10,1_o} & \dots & \mathbb{M}_{10,9_o} \\ & \dots & & \end{array}\right]}_{\mathbb{M}} \left[\begin{array}{c} \mathbf{V}_{(0)} \\ \mathbf{V}_{(1)} \\ \mathbf{V}_{(2)} \\ \mathbf{V}_{(3)} \\ \mathbf{V}_{(4)} \\ \mathbf{V}_{(5)} \\ \mathbf{V}_{(6)} \\ \mathbf{V}_{(7)} \\ \mathbf{V}_{(8)} \\ \mathbf{V}_{(9)} \end{array}\right] = \underbrace{\left[\begin{array}{c} \mathcal{R}_{0_A} \\ \mathcal{R}_{0_E} \\ \\ \mathcal{R}_{1_A} \\ \mathcal{R}_{1_E} \\ \dots \\ \mathcal{R}_{9_A} \\ \mathcal{R}_{9_E} \\ \hline \\ \mathcal{R}_{10_o} \\ \dots \end{array}\right]}_{\mathcal{R}} \tag{3.59}$$

The first 20 equations enforce a point-wise consistency of the inter-cell recon-

structed solution with the in-cell reconstructed solutions, in the cell centers ($\mathbf{r}_{\mathrm{A}}$ and $\mathbf{r}_{\mathrm{E}}$). These are – two solution values, four slopes, six curvatures, and eight (reconstructed) third derivatives. In the generic representation, the individual equations used for the assemble are

$$\frac{\partial^{s+k}}{\partial \bar{x}^s \bar{y}^k}\mathbf{U}\left(\bar{x},\bar{y}\right) = \left[\begin{array}{c} \mathbf{V}_{(0)} \\ \mathbf{V}_{(1)} \\ \mathbf{V}_{(2)} \\ \mathbf{V}_{(3)} \\ \mathbf{V}_{(4)} \\ \mathbf{V}_{(5)} \\ \mathbf{V}_{(6)} \\ \mathbf{V}_{(7)} \\ \mathbf{V}_{(8)} \\ \mathbf{V}_{(9)} \end{array}\right]_{\mathrm{AE}} \frac{\partial^{s+k}}{\partial \bar{x}^s \bar{y}^k} \left[\begin{array}{c} \mathcal{T}_{(0)}\left(\bar{x},\bar{y}\right) \\ \mathcal{T}_{(1)}\left(\bar{x},\bar{y}\right) \\ \mathcal{T}_{(2)}\left(\bar{x},\bar{y}\right) \\ \mathcal{T}_{(3)}\left(\bar{x},\bar{y}\right) \\ \mathcal{T}_{(4)}\left(\bar{x},\bar{y}\right) \\ \mathcal{T}_{(5)}\left(\bar{x},\bar{y}\right) \\ \mathcal{T}_{(6)}\left(\bar{x},\bar{y}\right) \\ \mathcal{T}_{(7)}\left(\bar{x},\bar{y}\right) \\ \mathcal{T}_{(8)}\left(\bar{x},\bar{y}\right) \\ \mathcal{T}_{(9)}\left(\bar{x},\bar{y}\right) \end{array}\right] \tag{3.60}$$

Similar to the procedure used for the in-cell reconstruction, we collect the coefficients in the front of $\mathbf{V}_{(0,\ldots,9)}$ – these are the $\mathbb{M}_{\gamma,\alpha_\beta}$ in eq.(3.59), and stash all the rest into right-hand-side terms $\mathcal{R}_{\gamma_\beta}$. In this case, $\gamma, \alpha = 0, ..., 9$, and β =A, E.

The equations

$$\mathbb{M}_{10,0_{\mathrm{o}}}\mathbf{V}_{(0)} + \mathbb{M}_{10,1_{\mathrm{o}}}\mathbf{V}_{(1)} + ... + \mathbb{M}_{10,9_{\mathrm{o}}}\mathbf{V}_{(9)} = \mathcal{R}_{10_{\mathrm{o}}}, \quad \mathrm{o} = 0, ..., 9 \tag{3.61}$$

represent the "blended-in" weak statements, which both increase the accuracy and the robustness of the reconstruction. Similar to the in-cell reconstruction, we use Gaussian integration and the larger weights for these constraints (mostly, set to $\omega_{_w} = 10$).

The boundary conditions are implemented by imposing the values/derivatives/etc. at boundary nodes, incorporated into the eq.(3.59) – along the lines of the discussion for the in-cell least squares reconstruction. The statements from the "missing-neighbor" elements are, obviously, dropped.

Finally, after the non-square matrix $\mathbb{M}$ and the vector $\mathcal{R}$ are assembled, we

solve for

$$\begin{bmatrix} \mathbf{V}_{(0)} \\ \mathbf{V}_{(1)} \\ \mathbf{V}_{(2)} \\ \mathbf{V}_{(3)} \\ \mathbf{V}_{(4)} \\ \mathbf{V}_{(5)} \\ \mathbf{V}_{(6)} \\ \mathbf{V}_{(7)} \\ \mathbf{V}_{(8)} \\ \mathbf{V}_{(9)} \end{bmatrix} = \left(\mathbb{M}^{\mathrm{T}}\mathbb{M}\right)^{-1} \left(\mathbb{M}^{\mathrm{T}}\boldsymbol{\mathcal{R}}\right) \tag{3.62}$$

The matrix $\mathbb{M}$ and the vector $\mathcal{R}$ are scaled, and we use the Gauss-Jordan algorithm for the matrix inversion.

3.4.6 Weighted residuals

The weighted residual vector for each finite element can be expressed as

$$\begin{aligned} \boldsymbol{\mathcal{R}}_k = \partial_t \mathbf{U}_k + \mathcal{A}_k \int_{\Gamma_e} \left(\mathbf{F}_j - \mathbf{D}_j\right) n_j \frac{\mathcal{B}_k}{|\mathbb{J}|} d\Gamma - \\ -\mathcal{A}_k \int_{\Omega_e} \left[\left(\mathbf{F}_j - \mathbf{D}_j\right)\left(\partial_j \mathcal{B}_k - \frac{\mathcal{B}_k}{|\mathbb{J}|}\partial_j |\mathbb{J}|\right) + \mathbf{S}\mathcal{B}_k\right] \frac{1}{|\mathbb{J}|} d\Omega \end{aligned} \tag{3.63}$$

where k denotes the DoFs we solve for, and

$$\partial_j \mathcal{B}_k = \frac{\partial \mathcal{B}_k}{\partial \xi}\frac{\partial \xi}{\partial x_j} + \frac{\partial \mathcal{B}_k}{\partial \eta}\frac{\partial \eta}{\partial x_j} + \frac{\partial \mathcal{B}_k}{\partial \zeta}\frac{\partial \zeta}{\partial x_j}, \quad x_j = \{x, y, z\} \tag{3.64}$$

$$\partial_j |\mathbb{J}| = \frac{\partial |\mathbb{J}|}{\partial \xi}\frac{\partial \xi}{\partial x_j} + \frac{\partial |\mathbb{J}|}{\partial \eta}\frac{\partial \eta}{\partial x_j} + \frac{\partial |\mathbb{J}|}{\partial \zeta}\frac{\partial \zeta}{\partial x_j} \tag{3.65}$$

The discrete forms of these residuals will appear in the non-linear solver, as described in Chapter 4.

The *face*- and *domain*-integrals – $\int_{\Gamma_e}$ and $\int_{\Omega_e}$ – correspondingly, are replaced by the weighted summation over the corresponding Gaussian points.

The hyperbolic fluxes $\mathbf{F}_j$ are computed using the piecewise-discontinuous solutions at each face Gaussian point, evaluated using the *in-cell reconstructed* solution profiles, Section 3.4.4, from each side of the face. Using the solutions from each side, $\mathbf{U}^{(\mathrm{g,L})}$ and $\mathbf{U}^{(\mathrm{g,R})}$, an approximate Riemann solver is applied, to compute the numerical flux. For the test cases with relatively large Mach numbers, we use the **Local Lax Friedrichs** (LLF) scheme [Tor99]:

$$\mathbf{F}_{\mathrm{j}}^{(\mathrm{g})} = \frac{1}{2}\left(\mathbf{F}_{\mathrm{j}}^{(\mathrm{g,L})} + \mathbf{F}_{\mathrm{j}}^{(\mathrm{g,R})} + \alpha_{\mathrm{LLF}}\lambda\left(\mathbf{U}^{(\mathrm{g,L})} - \mathbf{U}^{(\mathrm{g,R})}\right)\right) \tag{3.66}$$

where λ is the maximum eigenvalue, while α_{LLF} is the adjustable numerical diffusion coefficient – usually set to 1.

For the Navier-Stokes solver with very low Mach numbers, we either set the coefficient α_{LLF} to small numbers (say, for $M = 10^{-2}$ and $\mathrm{RDG}_{\mathrm{P_2P_3}} - \alpha_{\mathrm{LLF}} = 0.1$), or use the following "**incompressible**" variation:

$$\mathbf{F}_{\mathrm{j}}^{(\mathrm{g})} = \frac{1}{2}\left(\mathbf{F}_{\mathrm{j}}^{(\mathrm{g,L})} + \mathbf{F}_{\mathrm{j}}^{(\mathrm{g,R})} + |\mathbf{v}|_{max}\left(\mathbf{U}^{(\mathrm{g,L})} - \mathbf{U}^{(\mathrm{g,R})}\right)\right) \tag{3.67}$$

where $|\mathbf{v}|_{max}$ is the input parameter, corresponding to the maximum material velocity for the problem under consideration.

For evaluation of the diffusion fluxes, we utilize the *inter-cell reconstructed* solution profiles, Section 3.4.5, computing the corresponding numerical diffusion flux at each face Gaussian integration point. Here, we do not consider any jumps in material properties, though - we allow non-linearity of the diffusion coefficients (i.e., temperature-dependent).

The fluxes and source terms appearing in the domain integral of eq.(3.4.5) are evaluated using the *in-cell reconstructed* solution profiles. Because of using of the in-cell reconstructed solution, there is no need for variations/modifications as discussed in [LvL09], referred to as RDG-1x and RDG-2x (once/twice-partially-integrated), including additional extra boundary terms. Rather – our approach is along the lines of the RDG-1x (once-partially-integrated), which offers an easy implementation of non-linear diffusion operators.

Finally, time discretization operators and related non-linear solver and preconditioning will be discussed in Chapter 4.

3.4.7 Pros and cons

As discussed in the motivation for moving to the orthogonal basis functions, the current formulation is generally better-conditioned (and more diagonally-dominant), as compared to the (non-orthogonal) Taylor basis functions, developed in [LLNM09], [LLNM10c], [LLNM10b], [LLNM10a], [LLNC11], [LLN12], [LXL+12], [LXS+13], [XLFN14] and [XLN14]. This feature is very important for fully-implicit time discretization, used in this study. In particular, based on the numerous experimentation with the performance of this approach (see Chapter 5), we encounter no difficulty to use the high-order $\mathrm{RDG}_{\mathrm{P_2P_3}}$, even for very stiff physics, at very low Mach number conditions, on rather bad (highly distorted/stretched) meshes.

On the other hand, there is an extra cost during the reconstruction (both the in-cell and the inter-cell), due to the need to evaluate matrices, mapping from the degrees of freedom for our orthogonal basis functions, to the Taylor basis functions, which are still being utilized for the basic solution reconstruction procedures. When used within an implicit solver, the significant part of the computational cost is associated with linear steps (Krylov iterations) – both in terms of CPU time and memory storage (the size of the Krylov subspace). Thus, we tend to believe that an extra effort during the reconstruction when using the orthogonal basis functions is a worthwhile investment, enabling simulations of rather difficult physical problems, with the sought-after accuracy and robustness.

This Page is Intentionally Left Blank

Chapter 4

Fully-Implicit Time Discretization

In this chapter, we describe fully-implicit time discretization, Section 4.1, and related linear and non-linear solution algorithms, Sections 4.2 and 4.7.

4.1 Time discretization

4.1.1 Implicit time discretizations

An implicit time discretization of Eq.(2.1) can be written as

$$\begin{aligned} \vec{\mathcal{U}}^{[k]} &= \vec{\mathcal{U}}^{[n]} + \Delta t \sum_{r=1}^{k} \mathsf{a}_{kr} \vec{\mathcal{S}}\left(\vec{\mathcal{U}}^{[r]}\right), \quad k = 1, ..., s-1 \\ \vec{\mathcal{U}}^{[n+1]} &= \alpha \vec{\mathcal{U}}^{[n-1]} + \beta \vec{\mathcal{U}}^{[n]} + \Delta t \left(\mathsf{b}_0 \vec{\mathcal{S}}\left(\vec{\mathcal{U}}^{[n]}\right) + \sum_{r=1}^{s} \mathsf{b}_r \vec{\mathcal{S}}\left(\vec{\mathcal{U}}^{[r]}\right) \right) \end{aligned} \tag{4.1}$$

where s is the total number of implicit Runge-Kutta (IRK) stages, while a_{kr} and b_r are the stage and the main scheme weights, respectively.

- In the case of $\alpha = 0$, $\beta = 1$, $s = 1$, $\mathsf{b}_0 = 0$ and $\mathsf{b}_1 = 1$, Eq.(4.1) reduces to the first-order *Backward Euler* (BE_1) discretization.
- In the case of $\alpha = -\frac{\Delta t^2}{\Delta t_{n-1}\left(2\Delta t + \Delta t_{n-1}\right)}$, $\beta = \frac{\Delta t}{\Delta t_{n-1}} \frac{\Delta t + \Delta t_{n-1}}{2\Delta t + \Delta t_{n-1}}$, $s = 1$, $\mathsf{b}_0 = 0$ and $\mathsf{b}_1 = 1$, Eq.(4.1) reduces to the second-order *Backward Difference* (BDF_2) discretization.
- In the case of $\alpha = 0$, $\beta = 1$, $s = 1$, $\mathsf{b}_0 = \frac{1}{2}$ and $\mathsf{b}_1 = \frac{1}{2}$, it is the second-order *Crank-Nicholson* (CN_2) scheme.

4.1.2 Explicit, Singly Diagonally Implicit Runge-Kutta (ESDIRK)

A family of high-order IRK schemes recently developed by Carpenter et al. [BCVK02] [CKB+05] is particularly useful for all-speed flow solvers, since these schemes not only do not amplify any left-half-plane-(LHP)-scaled eigenvalues (*A-stability*), but also provide a complete damping of all eigenvalues including those at the limit $||z \to \infty||$ (*L-stability*). These IRK schemes are prescribed by $\mathsf{b}_0 = 0$ and the Butcher tableau of the following form:

$$\begin{array}{c|cccccc} 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ \mathsf{c}_2 & \mathsf{a}_{21} & \gamma & 0 & 0 & \ldots & 0 \\ \mathsf{c}_3 & \mathsf{a}_{31} & \mathsf{a}_{32} & \gamma & 0 & \ldots & 0 \\ \ldots & & & \ldots & & & \\ \mathsf{c}_{s-1} & \mathsf{a}_{(s-1)1} & \mathsf{a}_{(s-1)2} & \ldots & \ldots & \gamma & 0 \\ 1 & \mathsf{b}_1 & \mathsf{b}_2 & \mathsf{b}_3 & \ldots & \mathsf{b}_{(s-1)} & \gamma \\ \hline & \mathsf{b}_1 & \mathsf{b}_2 & \mathsf{b}_3 & \ldots & \mathsf{b}_{(s-1)} & \gamma \\ \hline & \hat{\mathsf{b}}_1 & \hat{\mathsf{b}}_2 & \hat{\mathsf{b}}_3 & \ldots & \hat{\mathsf{b}}_{(s-1)} & \hat{\mathsf{b}}_{(s)} \end{array} \tag{4.2}$$

where c_r denotes the point in time of the r^{th}-stage, $t^{[n]} + \mathsf{c}_r \Delta t$. Note that the first stage is explicit, and the diagonal elements for all stages $r > 1$ are the same, $\mathsf{a}_{rr} = \gamma$, which is why this family is called *"Explicit, Singly Diagonal Implicit Runge-Kutta" (ESDIRK)* in the literature. Note that the p^{th}-order ESDIRK$_p$ schemes allow to compute $(p-1)^{\text{th}}$-order solution, as

$$\vec{\mathcal{U}}^{[n+1]} = \vec{\mathcal{U}}^{[n]} + \Delta t \sum_{r=1}^{s} \hat{\mathsf{b}}_r \vec{\mathcal{S}}\left(\vec{\mathcal{U}}^{[r]}\right) \tag{4.3}$$

The coefficients of ESDIRK$_{3,4,5}$'s Butcher tableau were derived in [BCVK02, CKB+05, NTP+08], and given in Appendix B.

4.2 Newton-Krylov solver

Each stage of IRK eq.(4.1) requires solution of the non-linear system in the form

$$\vec{res}\left(\vec{\mathcal{X}}\right) = 0 \tag{4.4}$$

where

$$\vec{\mathcal{X}} = \left(\vec{\mathcal{U}}_1^{(k=0,\ldots,p)^{\mathsf{T}}}, \vec{\mathcal{U}}_2^{(k=0,\ldots,p)^{\mathsf{T}}}, \ldots, \vec{\mathcal{U}}_{\mathrm{N_{cells}}}^{(k=0,\ldots,p)^{\mathsf{T}}}\right)^{\mathsf{T}} \tag{4.5}$$

is a solution vector which includes all $(p+1)$ degrees of freedom for all variables in all $\mathrm{N}_{\mathrm{cells}}$ computational cells[1]. The residual vector $\vec{res}$ for each DoF at the cell $_{(\mathrm{c})}$ takes the form:

$$res_{\mathcal{U}_{\mathrm{c}}^{(k)}} = \mathcal{U}_{\mathrm{c}}^{(k)[rk]} - \mathcal{U}_{\mathrm{c}}^{(k)[n]} - \Delta t \sum_{r=1}^{rk} \mathsf{a}_{rk,r} \mathcal{S}_{\mathrm{c}}^{(k)} \left(\vec{\mathcal{X}}^{[r]} \right) \tag{4.6}$$

To solve eq.(4.4), we use the *Newton-Krylov* (NK) method [KK04]. Details of the used inexact Newton algorithm, Jacobian-free GMRES method, and preconditioning are discussed next.

4.3 Newton's method

We solve Eq.(4.6) with Newton's method, iteratively, as a sequence of linear problems defined by

$$\mathbb{J}^a \delta \vec{\mathcal{X}}^a = -\vec{res} \left(\vec{\mathcal{X}}^a \right) \tag{4.7}$$

The matrix $\mathbb{J}^a$ is the Jacobian of the a^{th} Newton's iteration and $\delta \vec{\mathcal{X}}^a$ is the update vector. Each $_{(i,j)^{\mathrm{th}}}$ element of the Jacobian matrix is a partial derivative of the i^{th} equation with respect to the j^{th} variable:

$$\mathbb{J}_{i,j} \equiv \frac{\partial res_i}{\partial \mathcal{X}_j} \tag{4.8}$$

The linear system Eq.(4.7) is solved for $\delta \vec{\mathcal{X}}^a$, and the new Newton's iteration value for $\vec{\mathcal{X}}$ is then computed as

$$\vec{\mathcal{X}}^{a+1} = \vec{\mathcal{X}}^a + \lambda^a \delta \vec{\mathcal{X}}^a \tag{4.9}$$

where λ^a is the step-length determined by a line search procedure [DS83], while $\delta \vec{\mathcal{X}}^a$ is the search direction. Newton's iterations on $\vec{\mathcal{X}}$ are continued until the convergence criterion

$$\left|\left| \vec{res} \left(\vec{\mathcal{X}}^a \right) \right|\right|_2 < \mathrm{tol}_{\mathrm{N}} \left|\left| \vec{res} \left(\vec{\mathcal{X}}^0 \right) \right|\right|_2 \tag{4.10}$$

is satisfied. The nonlinear tolerance $\mathrm{tol}_{\mathrm{N}}$ is varied from 10^{-3} to $\mathrm{tol}_{\mathrm{N}} = 10^{-8}$.

[1] It is instructive to emphasize that the reconstructed DoFs are not a part of the solution vector.

4.4 Krylov subspace iterations (GMRES)

The linear solver used in our code is the Arnoldi-based *Generalized Minimal RESidual method (GMRES)* [SS86]. It belongs to the general class of Krylov subspace iteration methods. These projection (Galerkin) or generalized projection (Petrov-Galerkin) methods [Saa03] are suitable for solving non-symmetric linear systems of the form eq.(4.7), using the Krylov subspace, $\mathbb{K}_j$,

$$\mathbb{K}_j = \text{span}\left(\vec{r}_0, \mathbb{J}\vec{r}_0, \mathbb{J}^2\vec{r}_0, ..., \mathbb{J}^{j-1}\vec{r}_0\right) \tag{4.11}$$

where $\vec{r}_0 = \mathbb{J}^a \delta\vec{\mathcal{X}}_0^a + r\vec{e}s\left(\vec{\mathcal{X}}^a\right)$. In GMRES, the Arnoldi basis vectors form a trial subspace out of which the m^{th}-iteration solution is constructed:

$$\delta\vec{\mathcal{X}}_m^a = \delta\vec{\mathcal{X}}_0^a + \mathfrak{k}_0\vec{r}_0 + \mathfrak{k}_1\mathbb{J}\vec{r}_0 + \mathfrak{k}_2\mathbb{J}^2\vec{r}_0 + ... + \mathfrak{k}_m\mathbb{J}^m\vec{r}_0 \tag{4.12}$$

where $(\mathfrak{k}_0, \mathfrak{k}_1, ..., \mathfrak{k}_m)$ are "coordinates" of the m^{th} trial solution in the Krylov subspace. As one can see, only matrix-vector products are required to create new trial vectors. The iterations are terminated based on a by-product (free) estimate of the residual that does not require explicit construction of intermediate residual vectors. This is a major advantage of GMRES over other Krylov methods. GMRES has a residual minimization property in the Euclidean norm. The major drawback of GMRES is that it requires the storage of all previous Arnoldi/(Krylov) basis vectors. This problem can be alleviated with efficient preconditioning.

4.5 Jacobian-free implementation

Since GMRES does not require individual elements of the Jacobian matrix $\mathbb{J}$, it never needs to be explicitely constructed. Instead only matrix-vector multiplications $\mathbb{J}\vec{\kappa}$ are needed, where $\vec{\kappa} \in (\vec{r}_0, \mathbb{J}\vec{r}_0, \mathbb{J}^2\vec{r}_0, ...)$ are Krylov vectors. Thus, *Jacobian-free implementations* are possible. The action of the Jacobian matrix can be approximated by Fréchet derivatives

$$\mathbb{J}\vec{\kappa} \approx \frac{r\vec{e}s\left(\vec{\mathcal{X}} + \varepsilon\vec{\kappa}\right) - r\vec{e}s\left(\vec{\mathcal{X}}\right)}{\varepsilon} \tag{4.13}$$

There are two approaches for choosing ε.

Brown & Saad. The first approach is due to [BS90]:

$$\varepsilon = \begin{cases} \epsilon_{\text{rel}} \frac{\vec{\mathcal{X}}^{\mathsf{T}} \vec{\kappa}}{||\vec{\kappa}||_2^2} & \text{if } \left| \vec{\mathcal{X}}' \vec{\kappa} \right| > \mathcal{X}_{\min} ||\vec{\kappa}||_1 \\ \epsilon_{\text{rel}} \mathcal{X}_{\min} \text{Sign} \left(\vec{\mathcal{X}}^{\mathsf{T}} \vec{\kappa} \right) \frac{||\vec{\kappa}||_1}{||\vec{\kappa}||_2^2} & \text{otherwise} \end{cases} \tag{4.14}$$

Pernice & Walker. The second approach is taken from [PW98]:

$$\varepsilon = \frac{\epsilon_{\text{rel}} \sqrt{1 + \left|\left| \vec{\mathcal{X}} \right|\right|}}{||\vec{\kappa}||} \tag{4.15}$$

The only control parameter is ϵ_{rel}. Note that for the entire linear iterative process $\vec{\mathcal{X}}$ does not change. Therefore, $\sqrt{1 + \left|\left| \vec{\mathcal{X}} \right|\right|}$ need be computed only once.

With the Jacobian-free formulation, the work associated with forming the Jacobian matrix and its storage can be eliminated, which is a significant saving of both CPU time and storage, provided that the number of Krylov vectors is kept small (see Section 4.7). Moreover, in many non-linear applications, the Jacobian matrix is not available due to size and complexity.

When used with (incomplete) factorization as a preconditioning techniques, explicit form of (approximate) Jacobian can be computed using eq.(4.13).

4.6 Inexact Newton

One important modification to Newton's method employed here is called an *inexact Newton's method* [KK04]. The term "inexact" refers to the accuracy of the iterative linear solver. The basic idea is that the linear system must be solved to a tight tolerance only when the added accuracy matters – i.e., when it affects the convergence of the Newton's iterations. This is accomplished by making the convergence of the linear residual proportional to the non-linear residual:

$$\left|\left| \mathbb{J}^a \delta \vec{\mathcal{X}}_m^a + r\vec{e}s \left(\vec{\mathcal{X}}^a \right) \right|\right| \leq \nu_a \left|\left| r\vec{e}s \left(\vec{\mathcal{X}}^a \right) \right|\right| \tag{4.16}$$

By default, ν_a is a constant (in most computations of the present study we use $\nu_a = 10^{-3}$). Alternatively, one can invoke the algorithm by Eisenstat and Walker [EW96], which computes ν_a at each step of the nonlinear solver.

4.7 Preconditioning

Because GMRES (Section 4.4) stores all of the previous Krylov vectors, it is necessary to keep the number of iterations relatively small, to prevent the storage and CPU time from becoming prohibitive. This is accomplished by preconditioning the linear system. *Preconditioning is a transformation of the original linear system into one with the same solution, but is easier to solve with an iterative solver* [Saa03]. We are using the right-preconditioned form of the linear system,

$$\underbrace{\mathbb{J}^a\mathbb{P}^{-1}}_{\hat{\mathbb{J}}}\underbrace{\mathbb{P}\delta\vec{\mathcal{X}}^a}_{\delta\vec{\mathcal{Y}}^a} = -r\vec{e}s\left(\vec{\mathcal{X}}^a\right) \tag{4.17}$$

where $\mathbb{P}^{-1}$ approximates $\mathbb{J}^{-1}$. The right-preconditioned version of Eq.(4.13) is

$$\mathbb{J}\mathbb{P}^{-1}\vec{\kappa} \approx \frac{r\vec{e}s\left(\vec{\mathcal{X}} + \varepsilon\mathbb{P}^{-1}\vec{\kappa}\right) - r\vec{e}s\left(\vec{\mathcal{X}}\right)}{\varepsilon} \tag{4.18}$$

This operation is applied once per GMRES iteration, in two steps:

I. `Preconditioning: approximately solve` $\delta\vec{\mathcal{Y}}^a = \mathbb{P}^{-1}\vec{\kappa}$

II. `Compute matrix-free product:` $\mathbb{J}\delta\mathcal{Y}^a \approx \frac{r\vec{e}s\left(\vec{\mathcal{X}}^a+\varepsilon\delta\vec{\mathcal{Y}}^a\right)-r\vec{e}s\left(\vec{\mathcal{X}}^a\right)}{\varepsilon}$

Finding a good preconditioner is often a combination of art, science, and intuition. A mathematically good preconditioner should efficiently cluster the eigenvalues of the iteration matrix [Saa03, KK04]. A preconditioner can also be defined as any subsidiary *approximate solver* that is combined with an outer iteration technique (e.g., multigrid, or one of the Krylov iteration solvers). One of the simplest and most popular ways of defining a preconditioner is to perform an incomplete lower-upper (ILU) factorization of the original matrix $\mathbb{J}$. A number of variations – ILU(k), ILUT, ILUS, ILUC, etc. – are discussed in [Saa03]. This is what we use in the present study.

A variety of ILU-based preconditioners are available through PETSC [BGMS97, BBG+01, BBE+04]. In this study, we use *block Jacobi* or *overlapping Additive Schwarz* methods, which are supported in PETSC in parallel. In some simulations, we used direct solvers – either PETSC's LU factorization (supported only

in serial), or superlu (parallel implementations). To utilize these preconditioners, we need to compute and store approximate Jacobian matrices. For this, we use PETSC's utilities with finite difference approximations and coloring algorithms (see [BGMS97, BBG+01, BBE+04] for details).

PBP. An important class of preconditioners for the JFNK method is referred to as *Physics-Based-Preconditioning (PBP)* or PDE-based [KK04]. The motivation behind this approach is that there exist numerous legacy operator-split algorithms to solve nonlinear systems. These algorithms were developed with insight into physical time scales of the problem. A direct benefit of this insight – a reduced implicit system, or a sequence of segregated semi-implicit solvers can be applied, instead of attempting to solve the fully-coupled system. Relevant fluid dynamics examples include the semi-implicit all-speed-flow *Implicit Continuous-fluid Eulerian (ICE)* algorithm [HA71], the semi-implicit incompressible-flow *SIMPLE* [Pat80] and the *Projection* algorithms [Cho67]. While PBP is beyond the scope of the present study, we believe this is the way to go in the future.

4.8 Preconditioning with primitive variable formulation

While the residue function in the MWR are written to satisfy underlying **conservation laws**, eq.(3.1), conservative variables $\mathbf{U}$ are often a poor choice (see discussion in Section 2.2.3), as the Jacobian matrix

$$\mathbb{J}_{i,j} \equiv \frac{\partial res_i}{\partial \mathbf{U}_j}$$

might be extremely ill-conditioned. Instead, one can (should) solve for another (mathematically equivalent) set of unknowns, which will render a better conditioned system. We denote this set of unknowns as **primitive variables**, $\mathbf{V}$. Introducing transformation

$$\delta\mathbf{U} = \frac{\partial \mathbf{U}}{\partial \mathbf{V}}\delta\mathbf{V} \tag{4.19}$$

linear steps of Newton iterations can be written as:

$$\underbrace{\frac{\partial \vec{res}}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial \mathbf{V}}}_{\tilde{\mathbb{J}}_{i,j}\equiv\frac{\partial res_i}{\partial \mathbf{V}_j}}\delta\mathbf{V} = -\vec{res} \tag{4.20}$$

Note that with a proper choice of $\mathbf{V}$, the Jacobian

$$\tilde{\mathbb{J}}_{i,j} \equiv \frac{\partial res_i}{\partial \mathbf{V}_j}$$

is better conditioned. This should be viewed as preconditioning, as it is conceptually similar to eq.(4.17). It is important to note that change of variables do not affect conservation, as residue functions are still written in the conservative form. Upon convergence of non-linear iterations, conservation is satisfied to the chosen tolerance level.

Chapter 5

Numerical Examples

In this chapter, we present numerical demonstrations of our algorithm performance on different test problems. We start with verification of m-consistency of the RDG reconstruction, in Section 5.1, followed by manufactured solution for non-linear heat conduction, Section 5.2, where we confirm high-order convergence rates for both space and time discretization. Next, we demonstrate performance of the algorithm for solving non-linear diffusion problem on highly stretched and distorted mesh, using Kershaw Z-mesh, in Section 5.3, investigating mesh imprinting effects. Fluid dynamics examples are presented in Sections 5.5 (manufactured solution), 5.6 (driven cavity flows) and 5.7 (vortex shedding).

5.1 On m-consistency

Linear consistency as a constraint required to achieve a desired level of accuracy in gradient calculation of finite-volume methods was discussed by Barth and Jesperson [BJ89]. This is a highly desired property of the reconstruction, which is not only the measure of accuracy and robustness, but also a feature which minimizes mesh imprint effects.

It is well known that the least-squares reconstruction of in-cell gradients (i.e., our $\mathrm{RDG}_{\mathrm{P_0P_1}}$) is linearly consistent. This means that linear field is reconstructed *exactly* (to round-off), on any mesh. Since our RDG is hierarchical, all $\mathrm{RDG}_{\mathrm{P_nP_{n+1}}}$ for $n > 0$ are linearly consistent as well, as verified in Table 5.1. The method however is not quadratically consistent on *irregular* meshes, Table 5.2. This is because the second derivatives in reference space are different from those in physical space

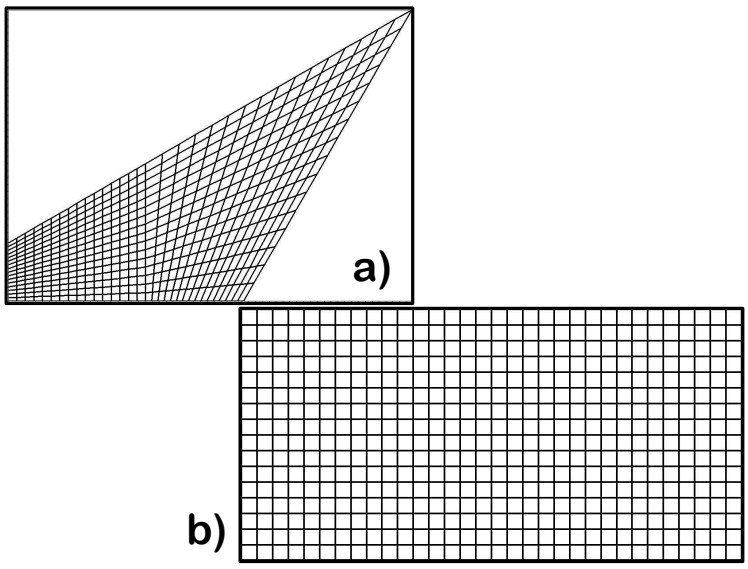


Fig. 5.1 : Meshes utilized for verification of m-consistency.

(due to transformation), and since the method is hierarchical in reference space, there is not quarantee that quadratic shape in physical space is reconstructed exactly. However, if the reconstructed function is quadratic in reference space, it is reconstructed exactly, as demonstrated using regular mesh[1], in Table 5.3. Similar, the $\mathrm{RDG}_{\mathrm{P_2P_3}}$ method is cubicly consistent in reference space, as expected, which is shown in Table 5.4. While both the $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and $\mathrm{RDG}_{\mathrm{P_1P_2}}$ are not cubically consistent, the reconstruction errors diminish with mesh refinement, with expected the second and the third-order, for the $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and $\mathrm{RDG}_{\mathrm{P_1P_2}}$ scheme, respectively.

[1] On regular mesh, reference and physical space are consistent, in terms of spatial derivatives.

Table 5.1 : On linear consistency ($m = 1$) of the reconstructed field $T(x, y) = 1 + x + y$, on irregular mesh shown in Figure 5.1a

METHOD	$\sqrt{N}$	$\mathcal{L}_2$-norm
$\mathrm{RDG}_{\mathrm{P_0P_1}}$	16	3.1580501711291E-14
		round-off
	32	3.1467415647531E-14
$\mathrm{RDG}_{\mathrm{P_1P_2}}$	16	3.2977841363630E-14
		round-off
	32	3.2182873435292E-14

Table 5.2 : On quadratic consistency ($m = 2$) of the reconstructed field $T(x, y) = 1 + x + y + \frac{x^2}{2} + xy + \frac{y^2}{2}$, on irregular mesh shown in Figure 5.1a

METHOD	$\sqrt{N}$	$\mathcal{L}_2$-norm
$\mathrm{RDG}_{\mathrm{P_0P_1}}$	128	2.4064E-5
		Rate: 2.031
	256	5.8810E-6
$\mathrm{RDG}_{\mathrm{P_1P_2}}$	128	9.3012E-8
		Rate: 3.002
	256	1.1612E-8

Table 5.3 : On quadratic consistency ($m = 2$) of the reconstructed field $T(x, y) = 1 + x + y + \frac{x^2}{2} + xy + \frac{y^2}{2}$, on regular mesh shown in Figure 5.1b

METHOD	$\sqrt{N}$	$\mathcal{L}_2$-norm
$\mathrm{RDG}_{\mathrm{P_0P_1}}$	128	2.9336E-5
		Rate: 2.096
	256	6.8633E-6
$\mathrm{RDG}_{\mathrm{P_1P_2}}$	128	5.3642839975098E-12
		Rate: 4.105, close to round-off
	256	3.1172059528318E-13
$\mathrm{RDG}_{\mathrm{P_2P_3}}$	128	5.1527272612064E-14
		Round-off
	256	5.1331277454087E-14

Table 5.4 : On cubic consistency ($m = 3$) of the reconstructed field $T(x, y) = 1 + x + y + \frac{x^2}{2} + xy + \frac{y^2}{2} + \frac{x^3}{6} + \frac{x^2y}{2} + \frac{xy^2}{2} + \frac{y^3}{6}$, on regular mesh shown in Figure 5.1b

METHOD	$\sqrt{N}$	$\mathcal{L}_2$-norm
$\mathrm{RDG}_{\mathrm{P_0P_1}}$	128	4.7060E-6
		Rate: 2.114
	256	1.0867E-6
$\mathrm{RDG}_{\mathrm{P_1P_2}}$	128	6.5828E-9
		Rate: 3.068
	256	7.8489E-10
$\mathrm{RDG}_{\mathrm{P_2P_3}}$	128	5.5017735726317E-14
		Round-off
	256	5.3830895500297E-14

5.2 Manufactured Solution for Non-Linear Heat Conduction

To test convergence in space and time for the problem with diffusion operator, the following solution is manufactured,

$$T(x,y) = A\cos(2\pi(x+bt))\sin(2\pi(y+ct)) \tag{5.1}$$

in 2D, and

$$T(x,y,z) = A\cos(2\pi(x+bt))\sin(2\pi(y+ct))\sin\left(\frac{\pi}{2}(z+dt)\right) \tag{5.2}$$

in 3D; where

$$A = 1 + a\sin(2\pi t) \tag{5.3}$$

and a, b, c, d are given constants.

In 2D, we used computational domain shown in Figure 5.2. There are six geometrical parameters, describing computational domain: X_0, X_1, Y_0, Y_1, α and β. We set

$$\begin{aligned} X_0 &= -1 \\ X_1 &= 1 \\ Y_0 &= -0.5 \\ Y_1 &= 0.5 \end{aligned}$$

while α and β are varied, to allow different levels of mesh stretching/distortion.

For 3D domain, we add three geometrical parameters – Z_0, Z_1 and γ, Figure 5.3. We fixed

$$\begin{aligned} Z_0 &= -0.4 \\ Z_1 &= 0.4 \end{aligned}$$

and allow γ to vary.

Manufactured solutions eqs.(5.1) and (5.2) correspond to a translating (with velocity $\mathbf{w}$) and oscillating (with amplitude a) thermal wave, as shown in Figures 5.4 and 5.5 for 2D, and in Figures 5.6 and 5.5 for 3D. In these simulations, we set

$$\begin{aligned} \alpha &= \beta = \gamma = 10^\circ \\ a &= \tfrac{1}{2} \\ \mathbf{w} &= (b,c,d) = \left(\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2}\right) \end{aligned}$$

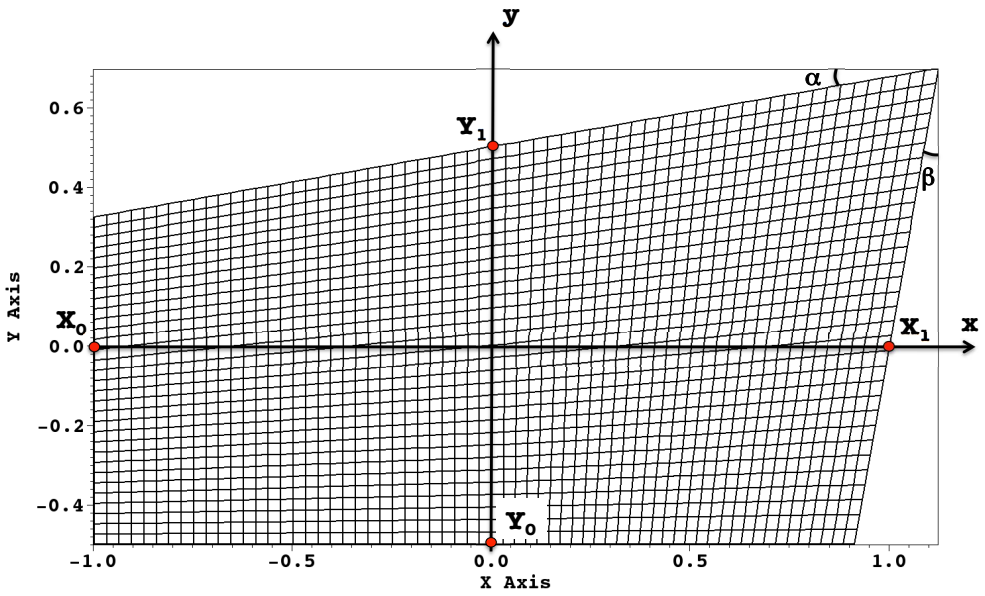


Fig. 5.2 : On domain and mesh setup for 2D manufactured solution test problem. $\alpha = \beta = 10°$.

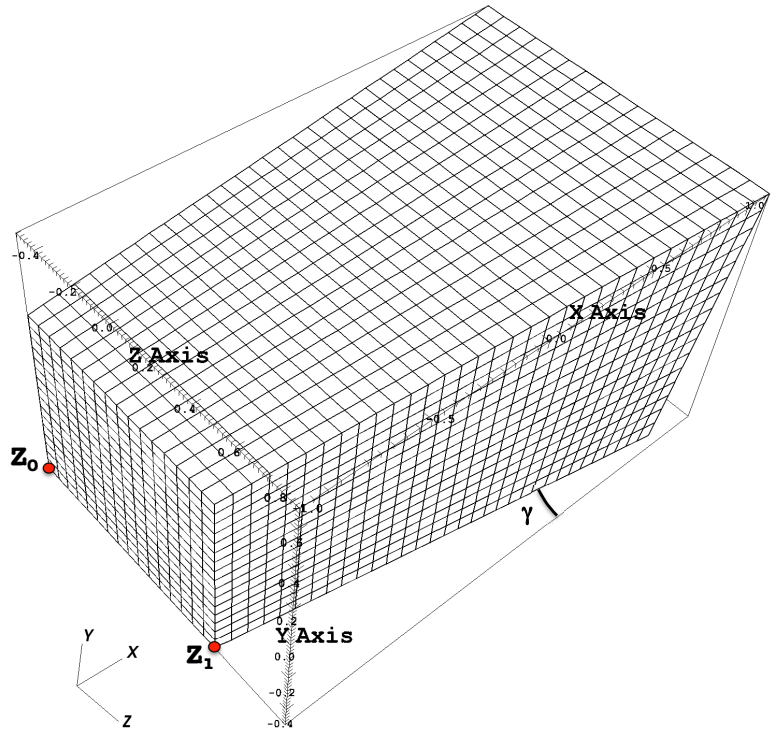


Fig. 5.3 : On domain and mesh setup for 3D manufactured solution test problem. $\alpha = \beta = \gamma = 10°$.

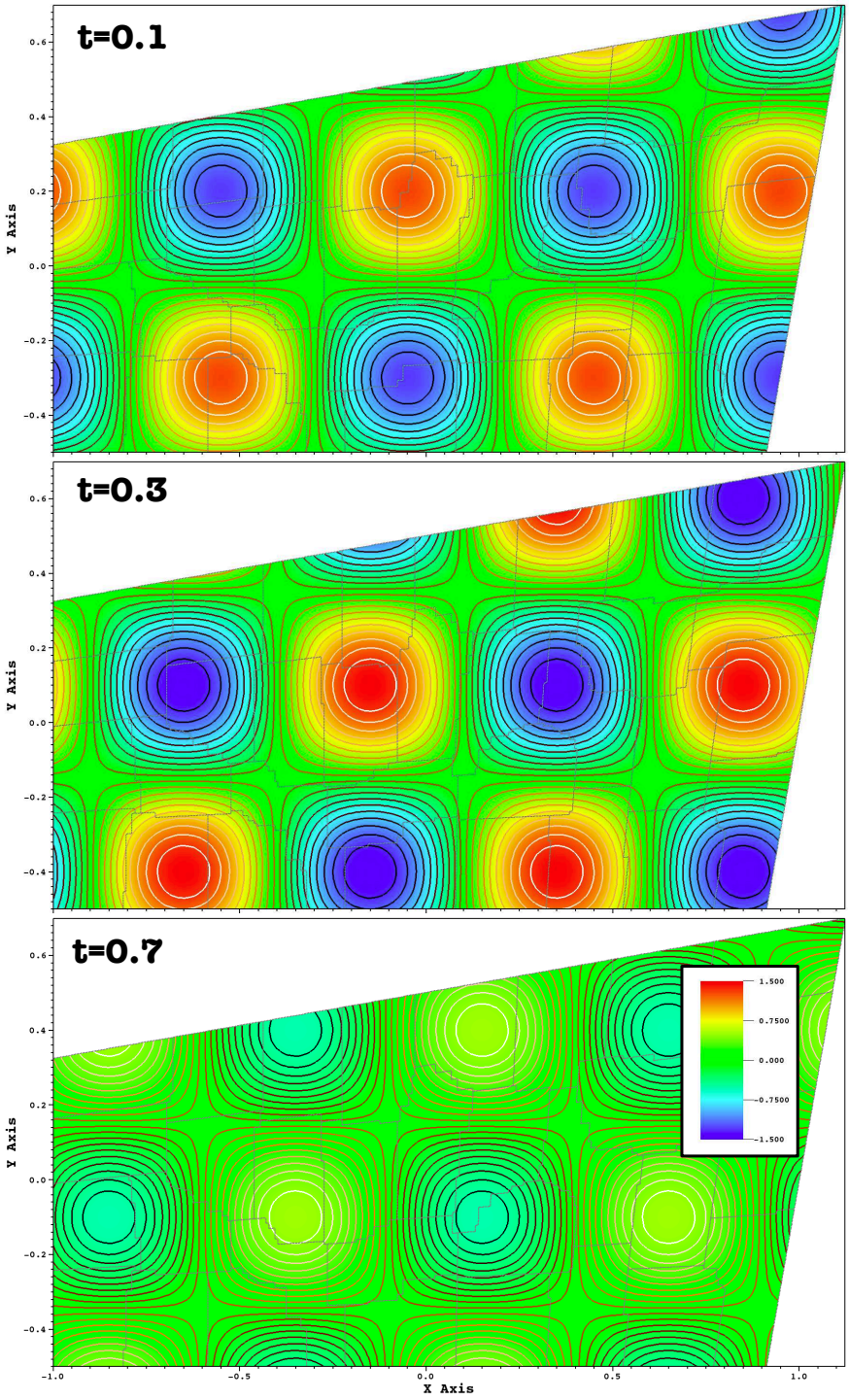


Fig. 5.4 : 2D manufactured temperature field as a function of time. Simulation with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ and ESDIRK_3 schemes, using $\Delta t = 0.1$ on domain with 32,762 elements, partitioned with 32 CPUs (partitioning is also shown).

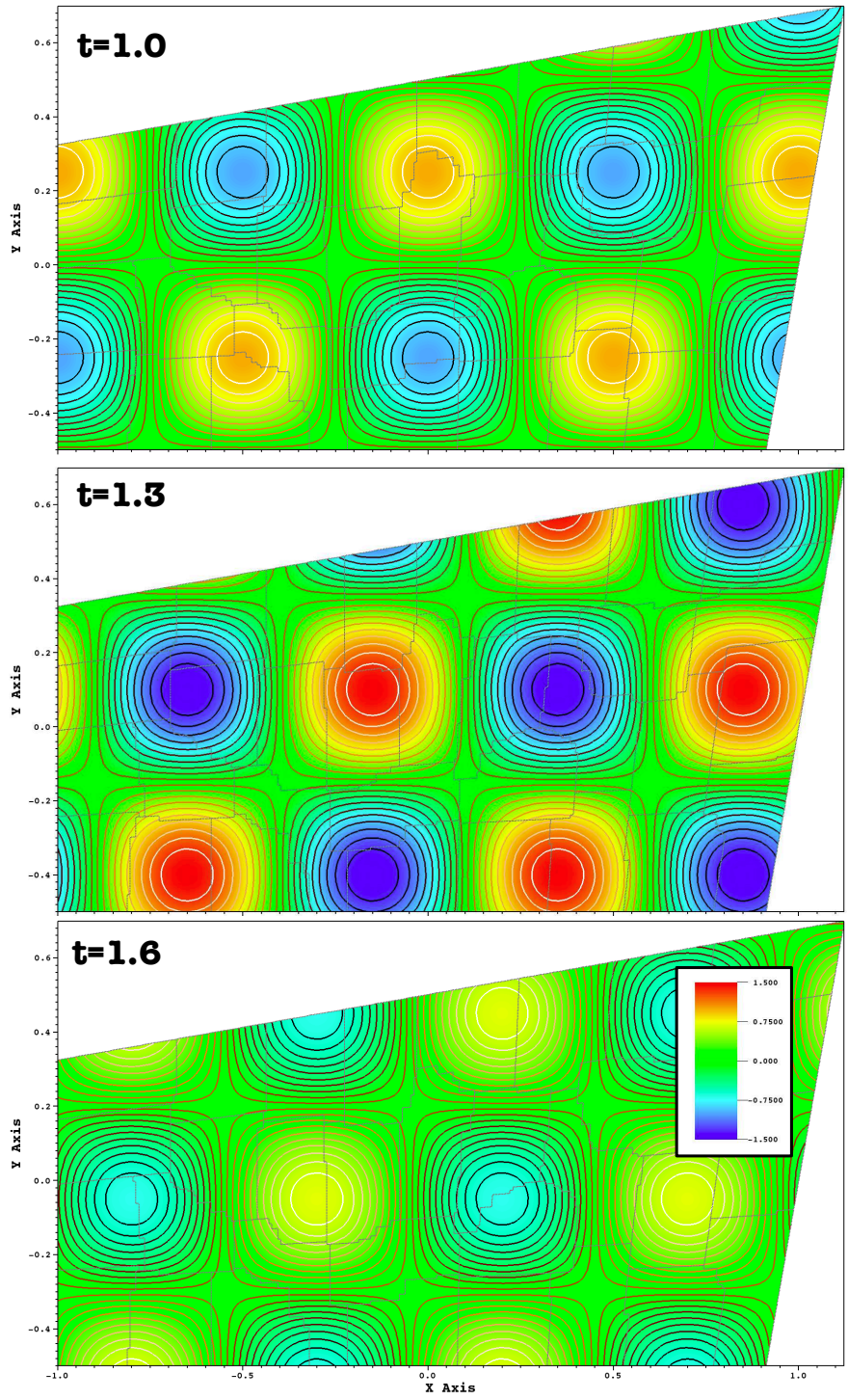


Fig. 5.5 : 2D manufactured temperature field as a function of time. Simulation with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ and ESDIRK_3 schemes, using $\Delta t = 0.1$ on domain with 32,762 elements, partitioned with 32 CPUs (continued).

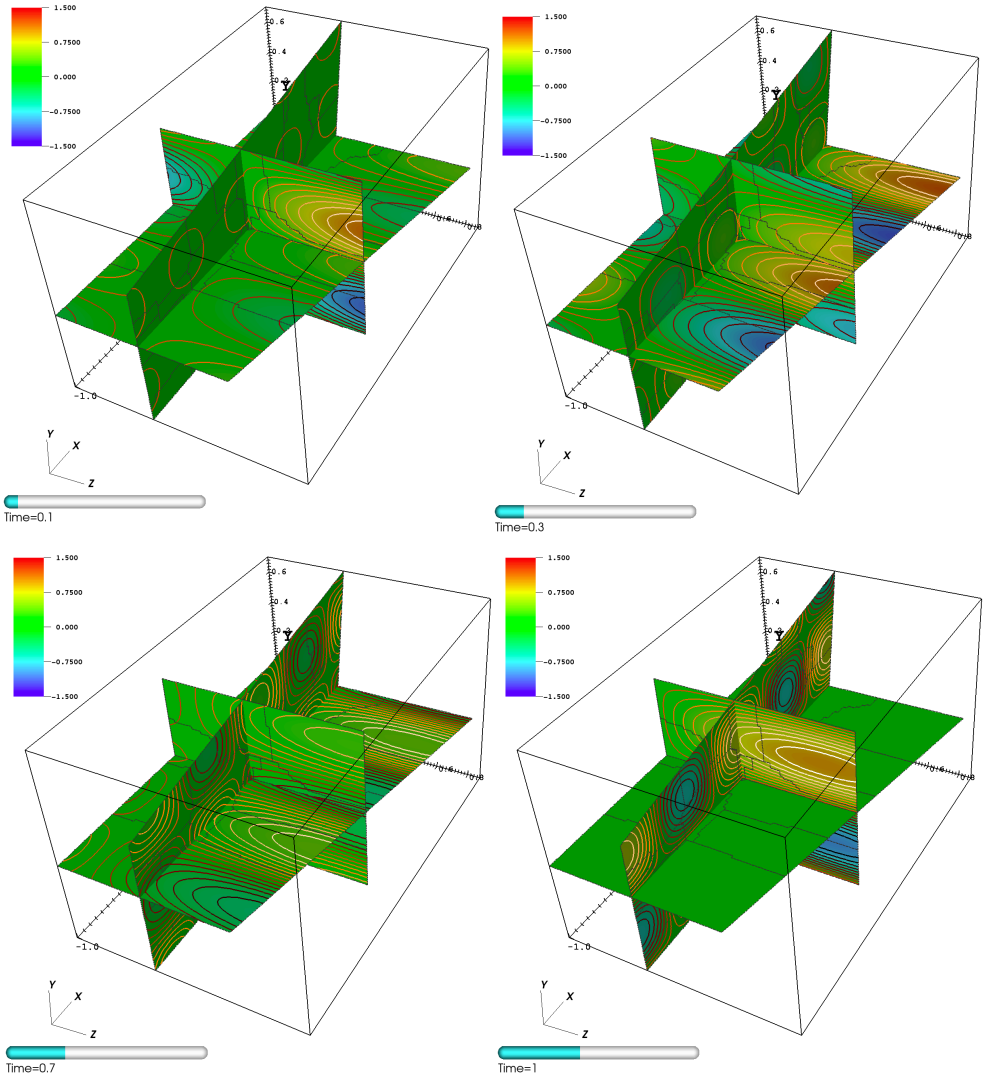


Fig. 5.6 : 3D manufactured temperature field as a function of time. Simulation with $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and ESDIRK_3 schemes, using $\Delta t = 0.1$ on domain with 524,288 elements, partitioned with 32 CPUs (partitioning is also shown).

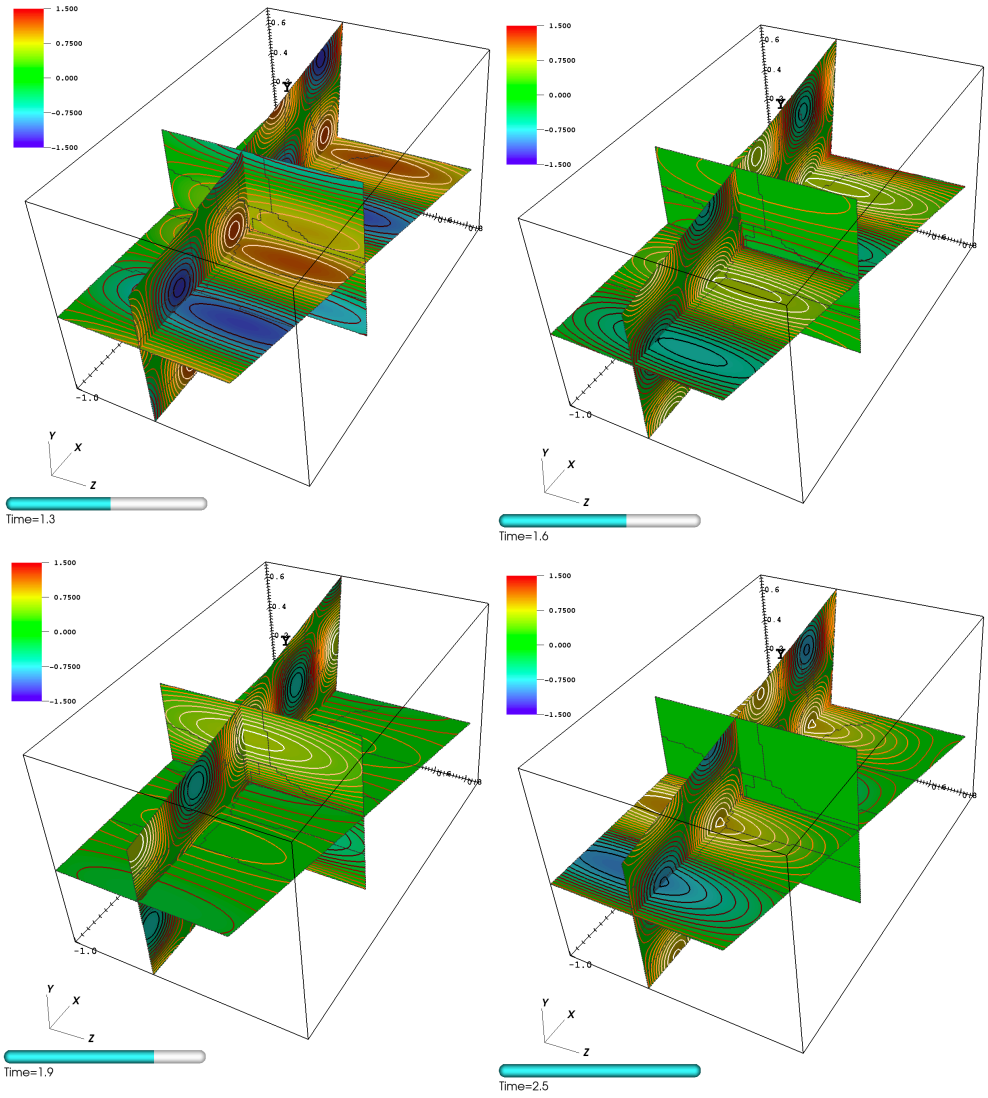


Fig. 5.7 : 3D manufactured temperature field as a function of time. Simulation with $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and ESDIRK_3 schemes, using $\Delta t = 0.1$ on domain with 524,288 elements, partitioned with 32 CPUs (continued).

Material properties are temperature dependent,

$$\begin{aligned} \rho(T) &= \rho_0 + \rho_1 T \\ \tilde{C}_v(T) &= \tilde{C}_{v_0} + \tilde{C}_{v_1} T \\ \kappa(T) &= \kappa_0 + \kappa_1 T \end{aligned} \tag{5.4}$$

Unless otherwise noted, we set $\rho_0 = 1$, $\rho_1 = 0$, $\tilde{C}_{v_0} = 1$, $\tilde{C}_{v_1} = 0.1$, $\kappa_0 = 1$ and $\kappa_1 = 0.1$, which makes both transient and diffusion operators quadratic. Also, we show results for temperature formulation (Section 2.1.4). (Specific internal energy and enthalpy formulations performed similarly).

To enforce manufactured solutions, source terms are added to the r.h.s. of eq.(2.8), derived using symbolic calculations in Mathematica.

In the following sections, we test performance of three space discretizations –

- $\mathrm{RDG}_{\mathrm{P_0P_1}}$,
- $\mathrm{RDG}_{\mathrm{P_1P_2}}$ and
- $\mathrm{RDG}_{\mathrm{P_2P_3}}$,

and six time discretizations –

- Backward Euler (BE),
- 2nd-order backward differencing (BDF_2),
- Crank-Nicholson (CN_2) and
- three implicit Runge-Kutta schemes ($\mathrm{ESDIRK}_{3,4,5}$).

For non-linear iterations, we use inexact Newton with backtracking line search, solved to the tolerance of 10^{-8}. To precondition GMRES, we use additive Schwarz (AS) combined with ILU_5 [BBE+04].

5.2.1 Time convergence

First, we test time convergence of our method, using the fourth-order-accurate $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh with 32,768 elements, Figure 5.8. With this scheme on

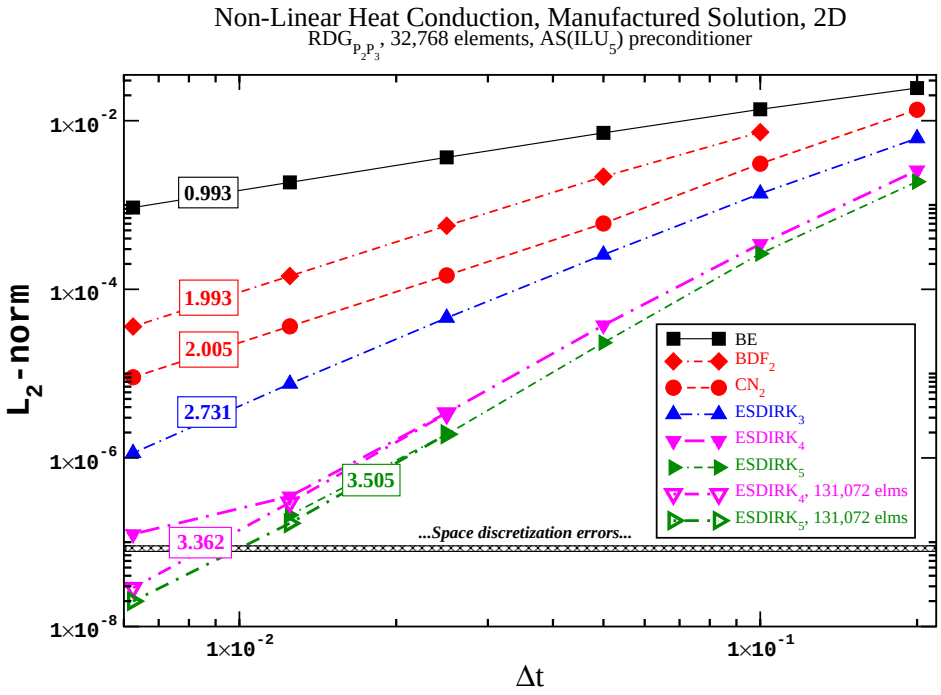


Fig. 5.8 : Time convergence of the temperature field, $t = 0.2$.

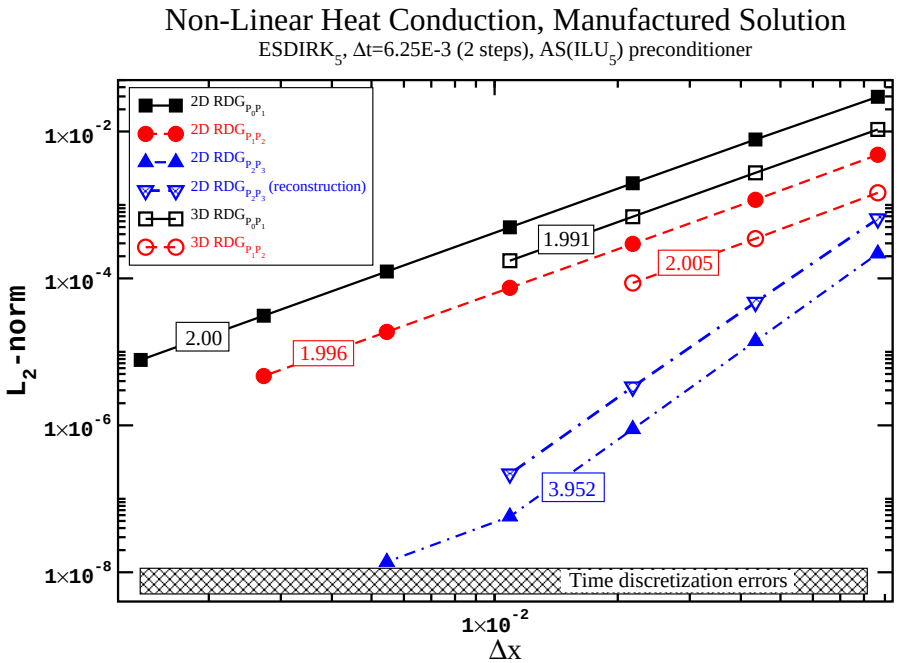


Fig. 5.9 : Spatial convergence of the temperature field.

this mesh, space discretization errors are on the order of $\sim 10^{-7}$, below time discretization errors of interest.

We note that the schemes up to the third order accuracy converge with rates close to theoretical. For higher-order $\text{ESDIRK}_{4,5}$, we could not attain theoretical the fourth- and the fifth-order convergence rate, most probably because time discretization errors are small, and becoming comparable with space discretization errors, at the asymptotic convergence range of time steps.

5.2.2 Space convergence

Next, we demonstrate convergence in space, by setting time discretization to ESDIRK_5 and $2 \times \Delta t = 0.00625$. With these, time discretization errors are at $\sim 10^{-8}$, which allows an accurate measurement of space convergence, Figure 5.9.

First, we note the $\text{RDG}_{\text{P}_0\text{P}_1}$ and $\text{RDG}_{\text{P}_2\text{P}_3}$ converge with rates close to theoretical – i.e., the second (for $\text{RDG}_{\text{P}_0\text{P}_1}$) and the fourth (for $\text{RDG}_{\text{P}_2\text{P}_3}$), in both 2D and 3D formulations. The third-order $\text{RDG}_{\text{P}_1\text{P}_2}$ however converged with only second order rate. This is because the third-order terms of inter-cell reconstruction dissappear during face flux integration, as discussed in Section 3.4.5. It is necessary to have even-order scheme to get consistent discretization of diffusion fluxes. Nevertheless, linear RDG $\text{RDG}_{\text{P}_1\text{P}_2}$ is more accurate than finite-volume $\text{RDG}_{\text{P}_0\text{P}_1}$, as its second-order errors are generally smaller, Figure 5.9.

It is instructive to note that the blended inter-cell recovery/reconstruction operator is more accurate than pure reconstruction (with strong statements in least-squares problem only), as discussed in Section 3.4.5.

In general, the fourth-order $\text{RDG}_{\text{P}_2\text{P}_3}$ is superior, as for the mesh resolution of practical range – it is orders of magnitude more accurate. For this test problem, we are getting similarly accurate solutions using 2,048 elements with $\text{RDG}_{\text{P}_2\text{P}_3}$, as compared to 2,097,152-element-mesh with $\text{RDG}_{\text{P}_0\text{P}_1}$. This is $\times 1,000$ saving in just mesh allocation, which compensates plenty for additional expenses of $\text{RDG}_{\text{P}_2\text{P}_3}$, due to $\times 6$ larger solution vector per element, per variable, and computational cost for residual evaluation on elements with high-order. Additional benefits emerge due to conditioning of linear iterations, as the elements are larger (smaller effective Fo numbers), and the total sizes of the solution vectors for linear algebra are significantly smaller.

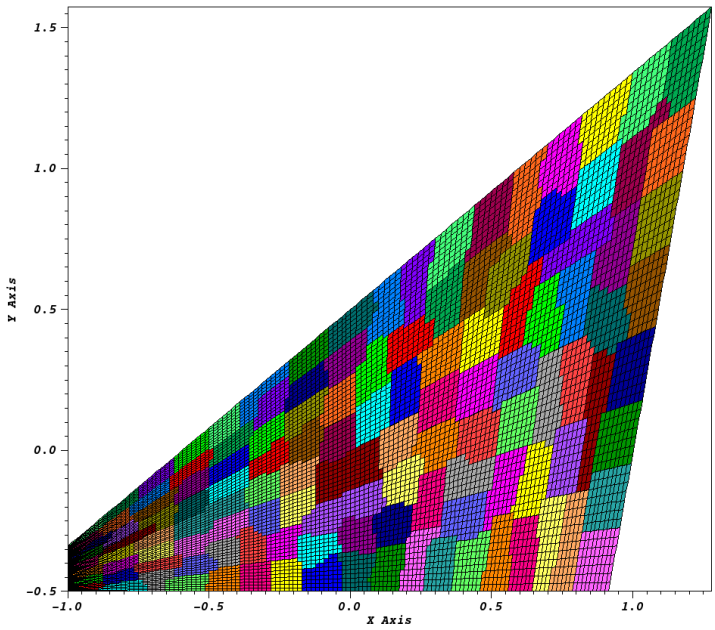


Fig. 5.10 : A sample of partitioning for 2D case, with 128 domains.

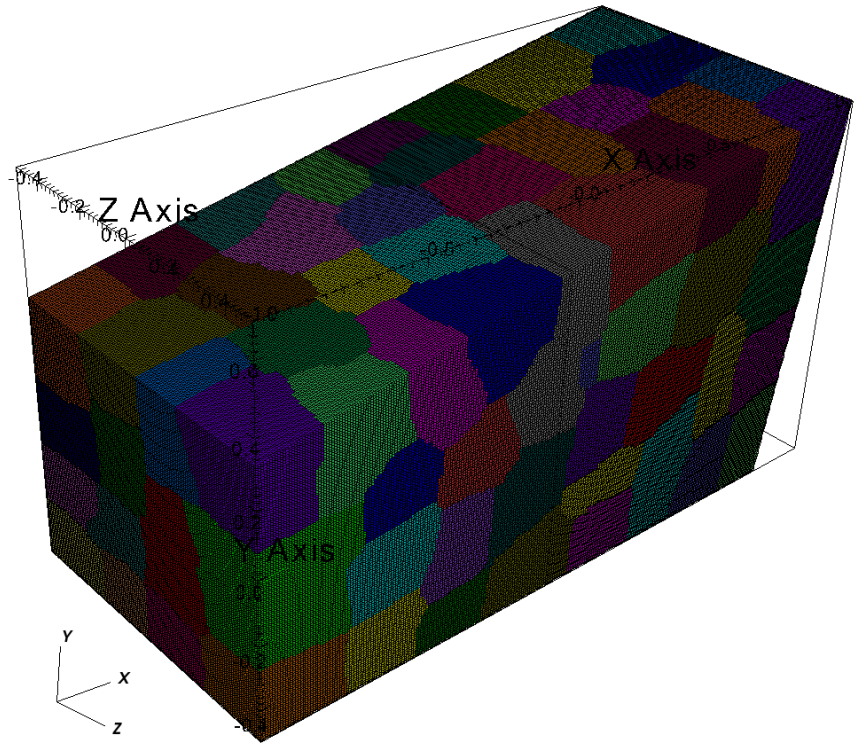


Fig. 5.11 : A sample of partitioning for 3D case, with 128 domains.

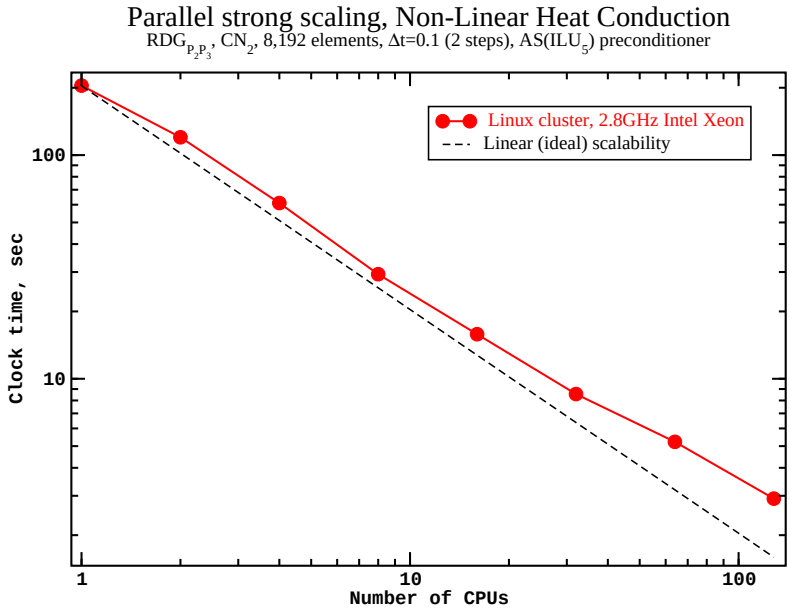


Fig. 5.12 : On strong parallel scalability of the algorithm, in 2D.

5.2.3 On parallel scalability

In this section, we show strong scalability of the algorithm, using the fourth-order-accurate $\mathrm{RDG}_{\mathrm{P_2P_3}}$. Even though high-order discretization is significantly more expensive than finite-volume $\mathrm{RDG}_{\mathrm{P_0P_1}}$, on the same mesh – these are mostly local costs, due to residual evaluation for each element. Thus, we can see very reasonable strong scalability even on very small mesh, as shown in Figure 5.12, when partitioned as shown in Figures 5.10 and 5.11.

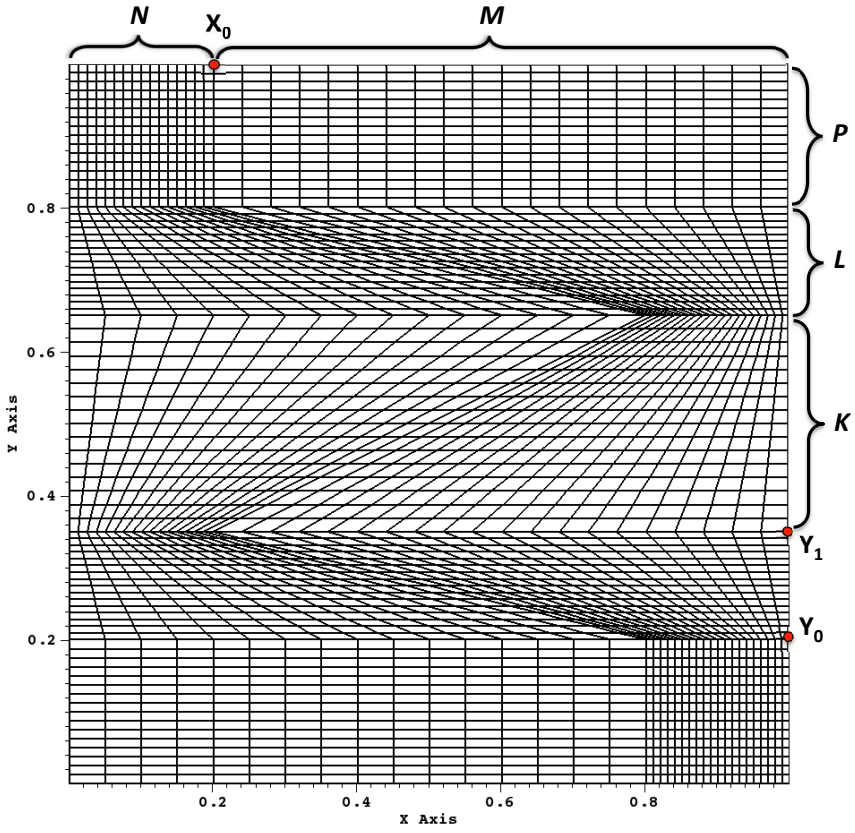


Fig. 5.13 : Kershaw Z mesh.

5.3 On mesh imprinting and related

To demonstrate the method's ability to compute diffusion operator through a mesh with highly skewed elements, we utilize the problem introduced by Kershaw in [Ker81]. In the computational domain of size 1×1, we generated a mesh with a distinct Z pattern, Figure 5.13. The mesh generation parameters are $X_0 = 0.2$, $Y_0 = 0.2$, and $Y_1 = 0.35$; with grid resolution set to $N = 16$, $L = 16$, $P = 16$ and $K = 20$. We solve non-linear heat conduction equation in temperature formulation, Section 2.1, with $\rho = 1$, $\tilde{C}_v = 1$ and temperature-dependent thermal conductivity,

$$\kappa(T) = \kappa_0 + \kappa_1 T$$

where $\kappa_0 = 1$ and $\kappa_1 = 0.1$. Initial temperature is set to 1. Neumann boundary conditions are applied at horizontal walls. For vertical walls, we use Dirichlet boundary conditions, with $T_L = 2$ at the left wall, and $T_R = 1$ at the right wall. Computations are performed with the third-order accurate ESDIRK$_3$ time discretization, with time step $\Delta t = 10^{-2}$, which corresponds to Fourier number $Fo \sim 2000$. Fourier number is defined as

$$Fo = \frac{\alpha \Delta t}{\Delta h^2} \tag{5.5}$$

where the thermal diffusivity α is computed as

$$\alpha = \frac{\kappa}{\rho C_v} \tag{5.6}$$

and Δh is the scale of the smallest elements. Nearly steady-state is achieved at $t = 1$.

Computational results are shown in Figures 5.14 and 5.15, for the second- and the fourth order-accurate space discretizations ($\mathrm{RDG}_{\mathrm{P_0P_1}}$ and $\mathrm{RDG}_{\mathrm{P_2P_3}}$). It can be clearly seen that no any mesh-imprint "chevron" patterns are observed, resulting in nearly perfect one-dimensional temperature field, with vertical straight isolines of temperature. The undesirable "chevron" patterns effects are attributted to many schemes for diffusion operator, as reported by Rebourcet in [Reb07]. We believe that negligible mesh-imprint features of our scheme are attributted to m-consistency, as discussed in Section 5.1.

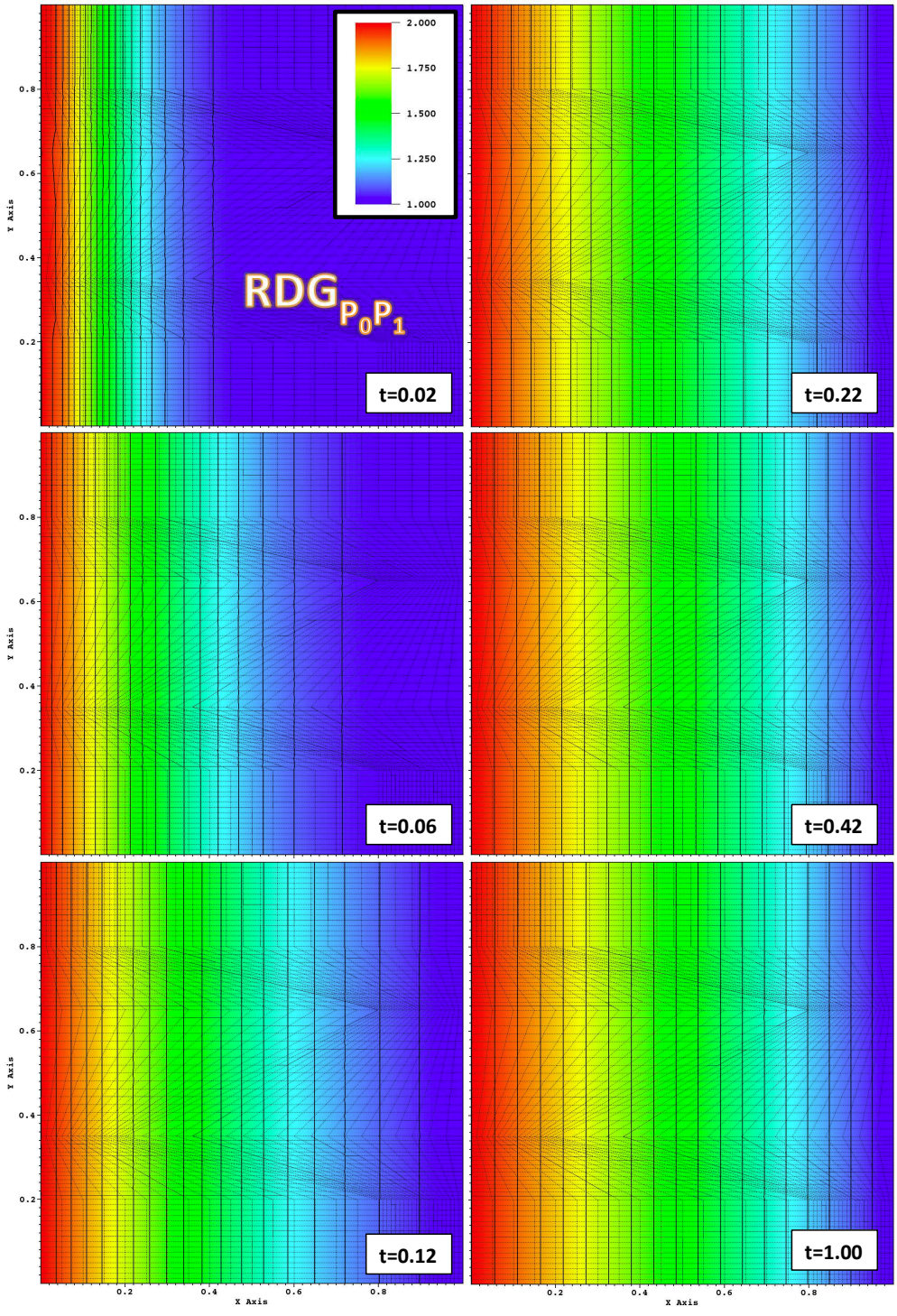


Fig. 5.14 : Temperature field as a function of time, for $\mathrm{RDG}_{\mathrm{P_0P_1}}$.

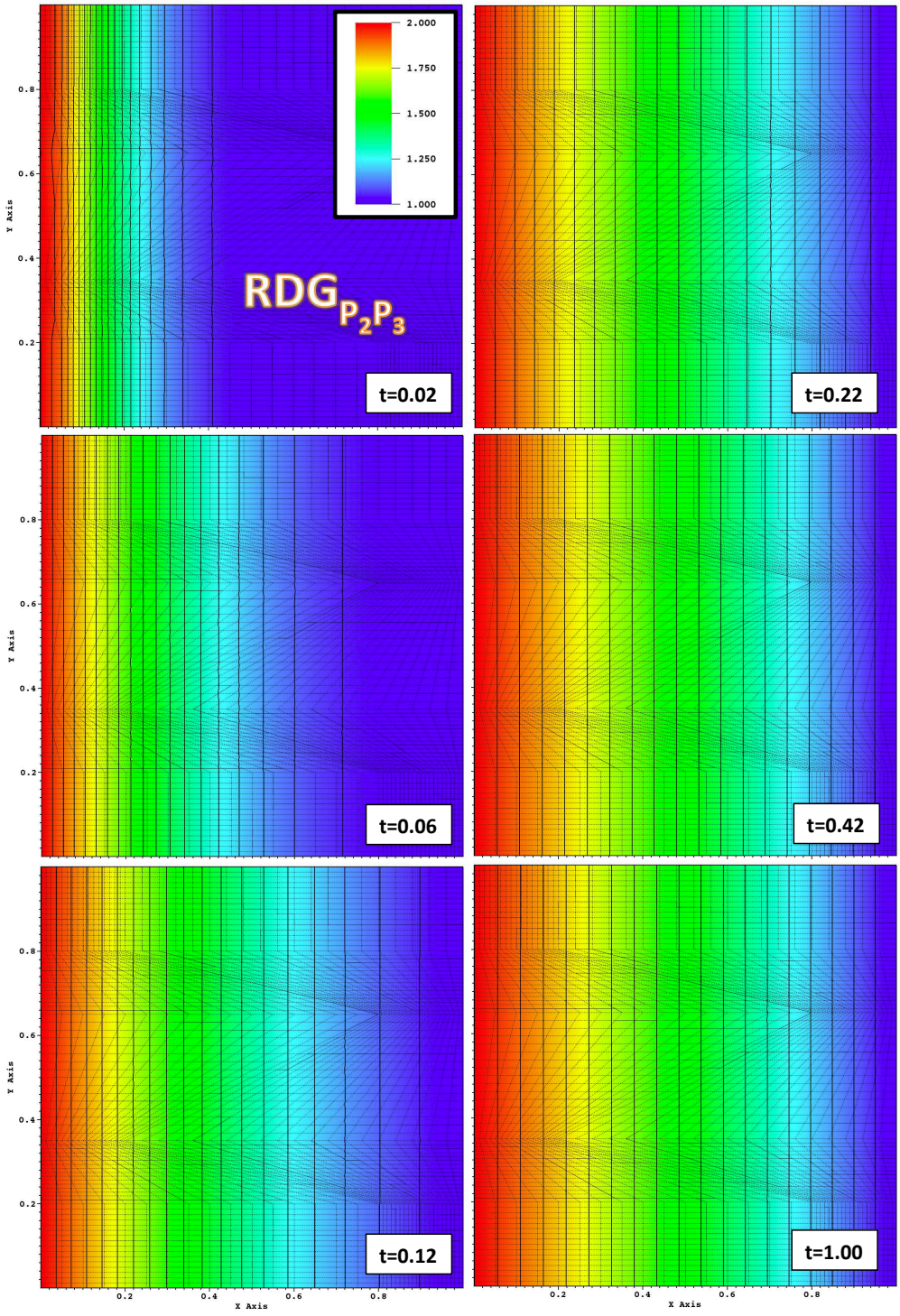


Fig. 5.15 : Temperature field as a function of time, for $\mathrm{RDG}_{\mathrm{P_2P_3}}$.

5.4 Melting of a stainless steel brick

Our next example is testing the method's performance for heat conduction with phase change. We consider melting of a large block made of stainless steel, subjected to heating from two side walls. Computational domain is described by Figure 5.2, with $\alpha = \beta = 20^{\circ}$. Initially, the whole block is at room temperature 300 K. At $t = 0$, we applied Dirichlet boundary conditions $T_{_L} = 1,800$ K from the left, and $T_{_R} = 2,000$ K from the right. Top and bottom walls are adiabatic (with Neumann bondary conditions applied). We solve energy equation (2.8), using temperature formulation, Section 2.1.4. The phase change model is described in Section 2.1.6.

The following properties are used for stainless steel:

$$\begin{aligned} \rho &= 7,753 \ \tfrac{\text{kg}}{\text{m}^3} \\ C_{_v} &= 486 \ \tfrac{\text{J}}{\text{kg}\cdot\text{K}} \\ \kappa &= 40 \ \tfrac{\text{W}}{\text{m}\cdot\text{K}} \\ T_{_S} &= 1,500 \text{ K} \\ T_{_L} &= 1,501 \text{ K} \\ \mathfrak{u}_{_f} &= 10^5 \ \tfrac{\text{J}}{\text{kg}} \end{aligned} \tag{5.7}$$

Equations are solved in non-dimensional form, Section 2.1.7, with

$$\begin{aligned} \bar{\rho} &= 7,753 \ \tfrac{\text{kg}}{\text{m}^3} \\ \bar{T} &= 1,000 \text{ K} \\ \bar{L} &= 1 \text{ m} \\ \bar{t} &= 1 \text{ sec} \\ \bar{C}_{_v} &= 486 \ \tfrac{\text{J}}{\text{kg}\cdot\text{K}} \end{aligned}$$

Thus, $\bar{\mathfrak{u}} = 4.86 \cdot 10^5 \ \frac{\text{J}}{\text{kg}}$ and $\bar{\kappa} = 3.77 \cdot 10^6 \ \frac{\text{W}}{\text{m}\cdot\text{K}}$; and dimensionless thermal conductivity is

$$\hat{\kappa} = 1.0616 \cdot 10^{-5}$$

Computational results are summarized in Figures 5.16-5.22. First, we show the history of the temperature field, in Figures 5.16 and 5.17. Simulations are

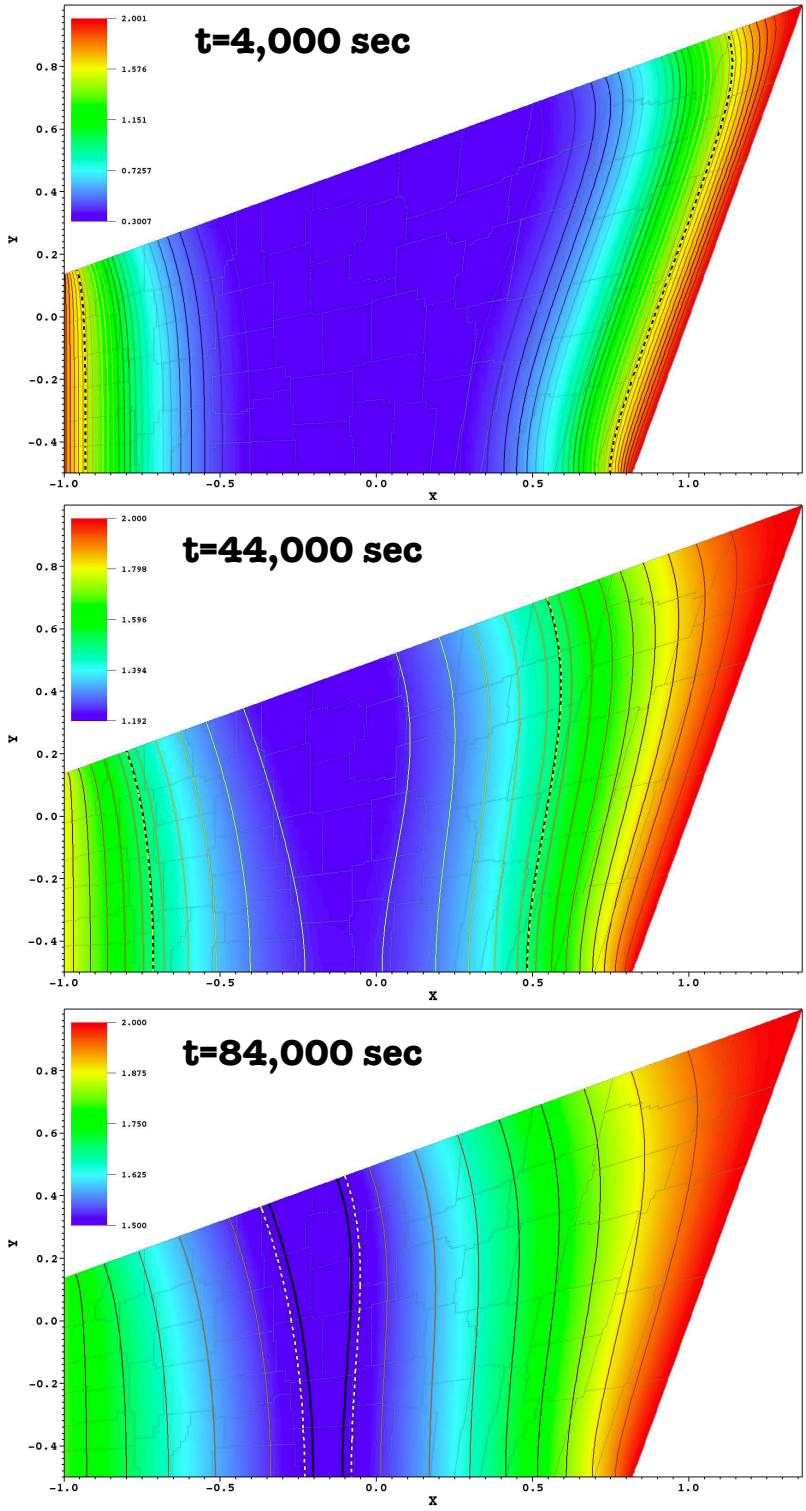


Fig. 5.16 : Temperature field and solidus/liquidus isolines as a function of time, for $\mathrm{RDG}_{\mathrm{P_2P_3}}$ with ESDIRK_3. 32,768 elements. $\Delta t = 2,000$ sec ($Fo \sim 850$). Temperature is scaled by $\bar{T} = 1,000$ K. CPU partitioning (domain decomposition) is also indicated as dotted lines.

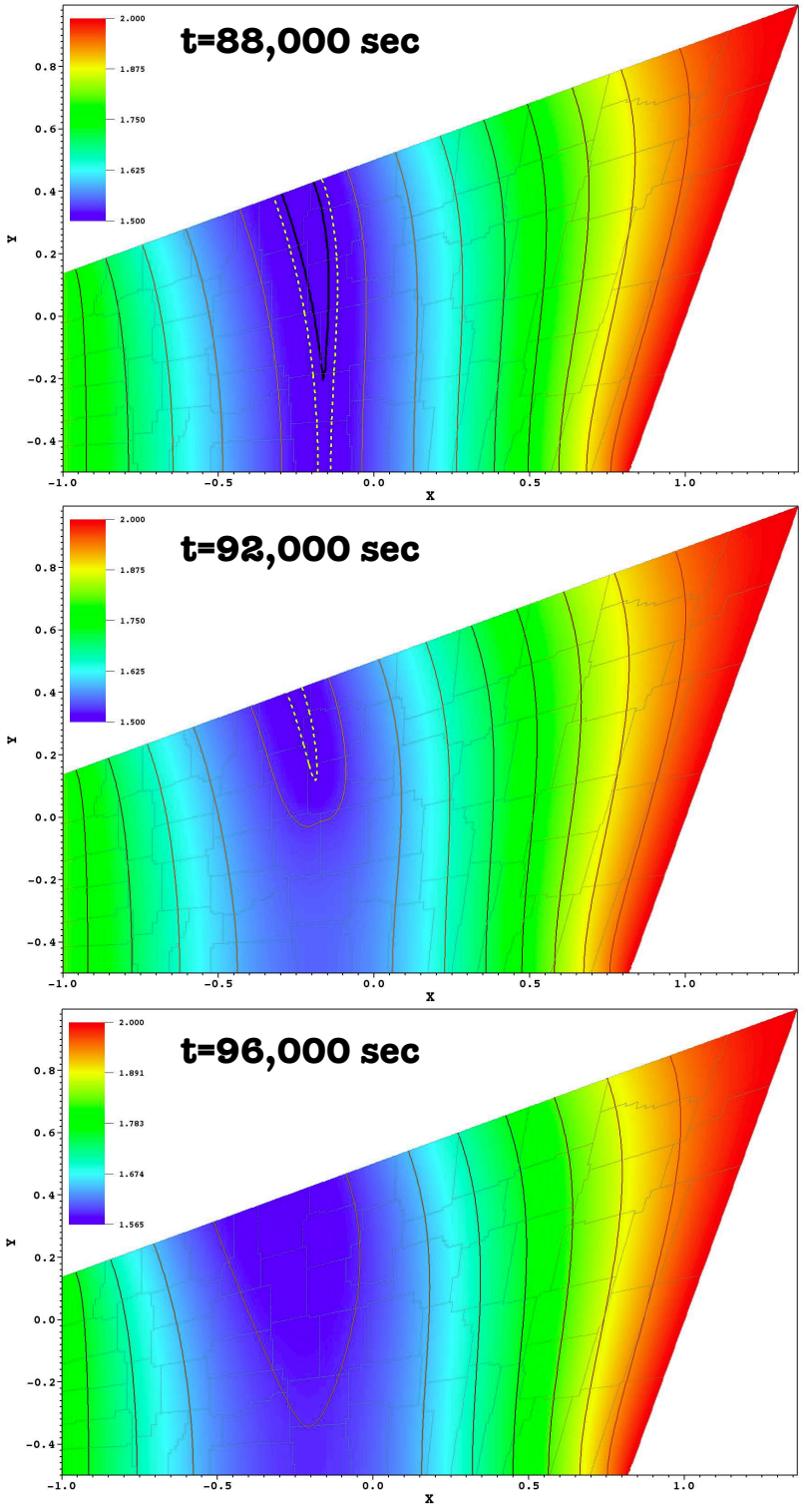


Fig. 5.17 : Temperature field and solidus/liquidus isolines as a function of time, for $\mathrm{RDG}_{\mathrm{P_2P_3}}$ with ESDIRK_3 (cont.).

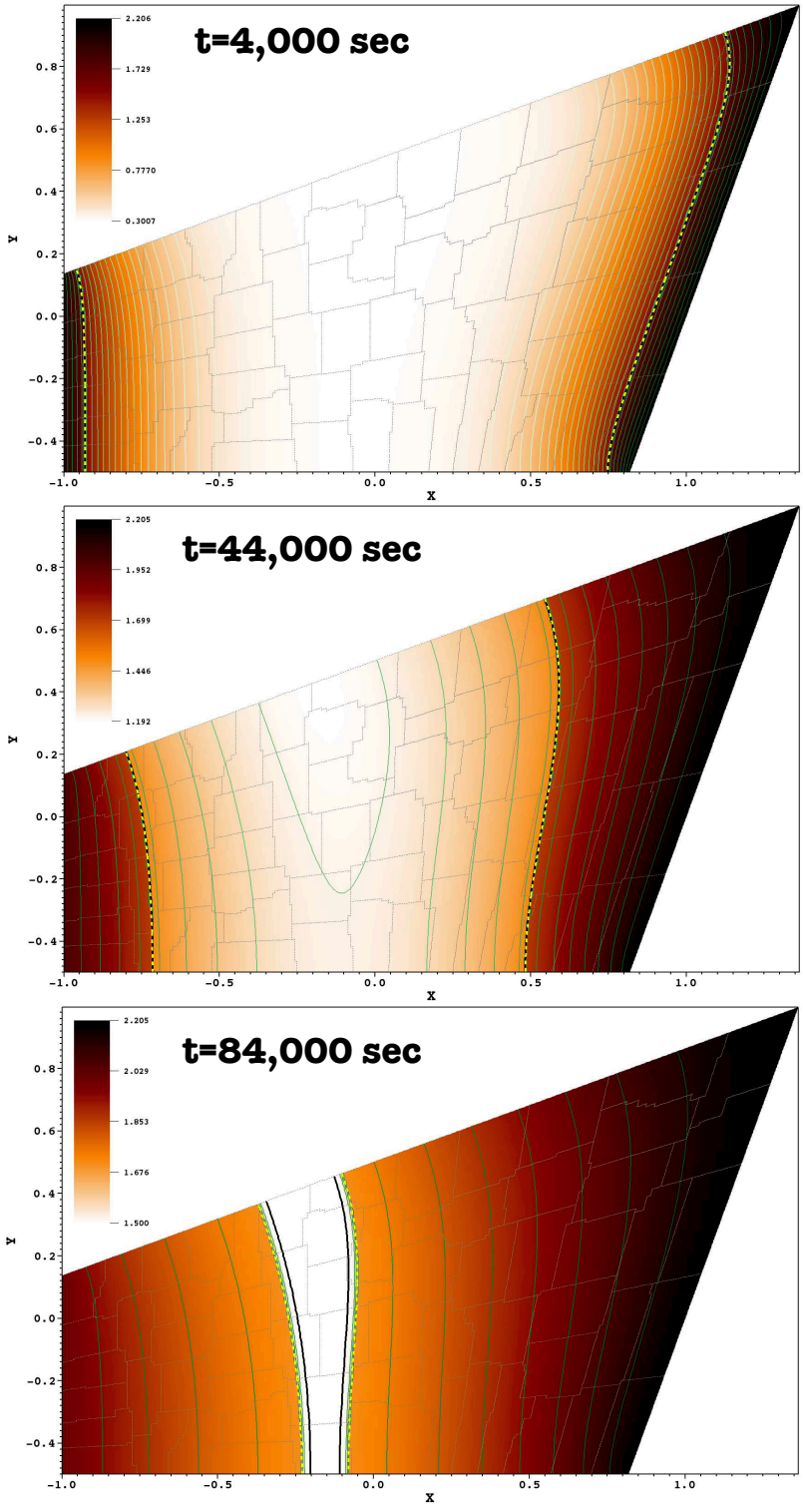


Fig. 5.18 : Specific internal energy field and solidus/liquidus isolines as a function of time, for $\mathrm{RDG}_{\mathrm{P_2P_3}}$ with ESDIRK_3. 32,768 elements. $\Delta t = 2,000$ sec ($Fo \sim 850$). Specific internal energy is scaled by $\bar{\mathfrak{u}} = 4.86 \cdot 10^5 \ \frac{\mathrm{J}}{\mathrm{kg}}$.

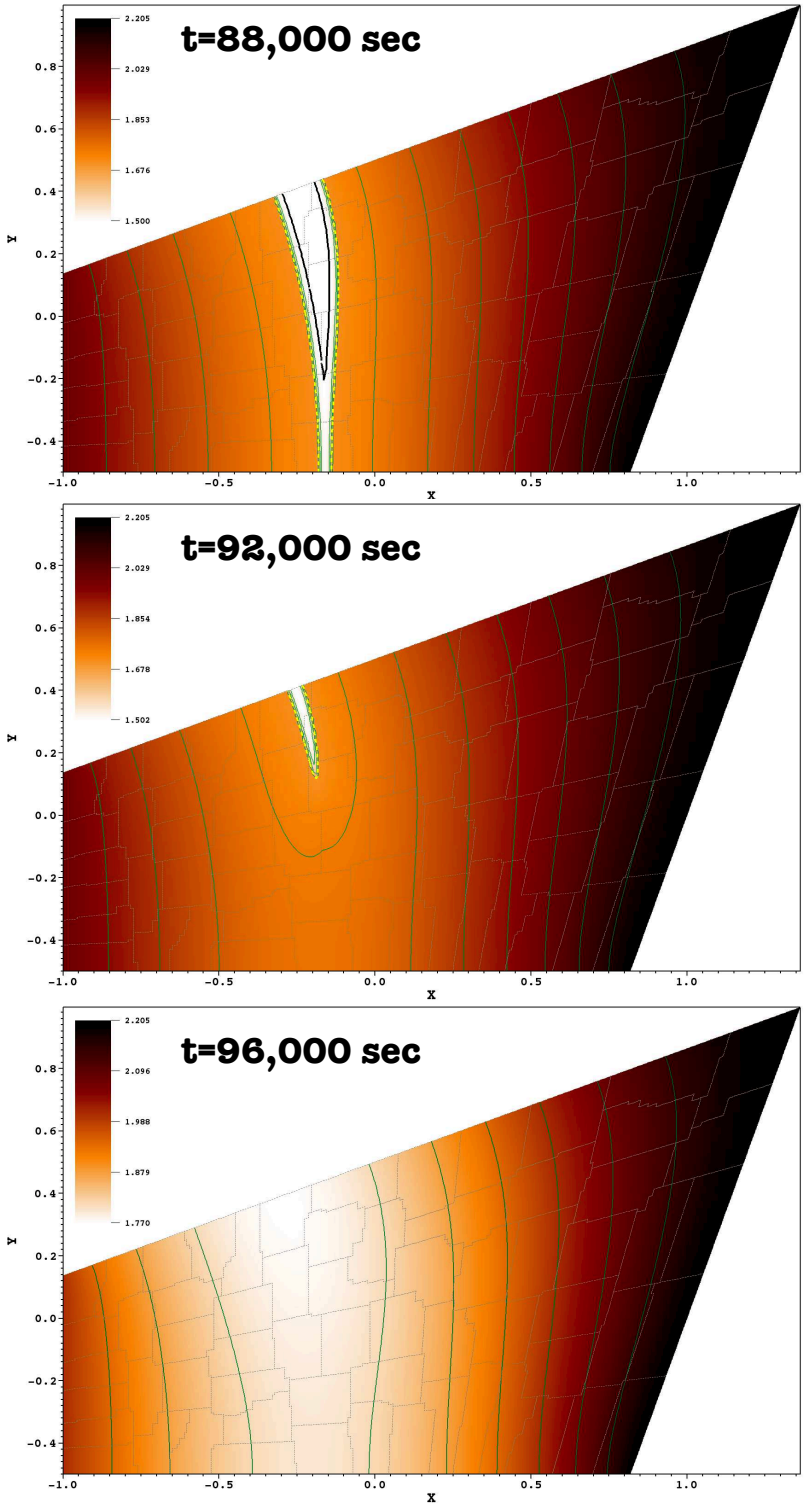


Fig. 5.19 : Specific internal energy field and solidus/liquidus isolines as a function of time, for $\mathrm{RDG}_{\mathrm{P_2P_3}}$ with ESDIRK_3 (cont.).

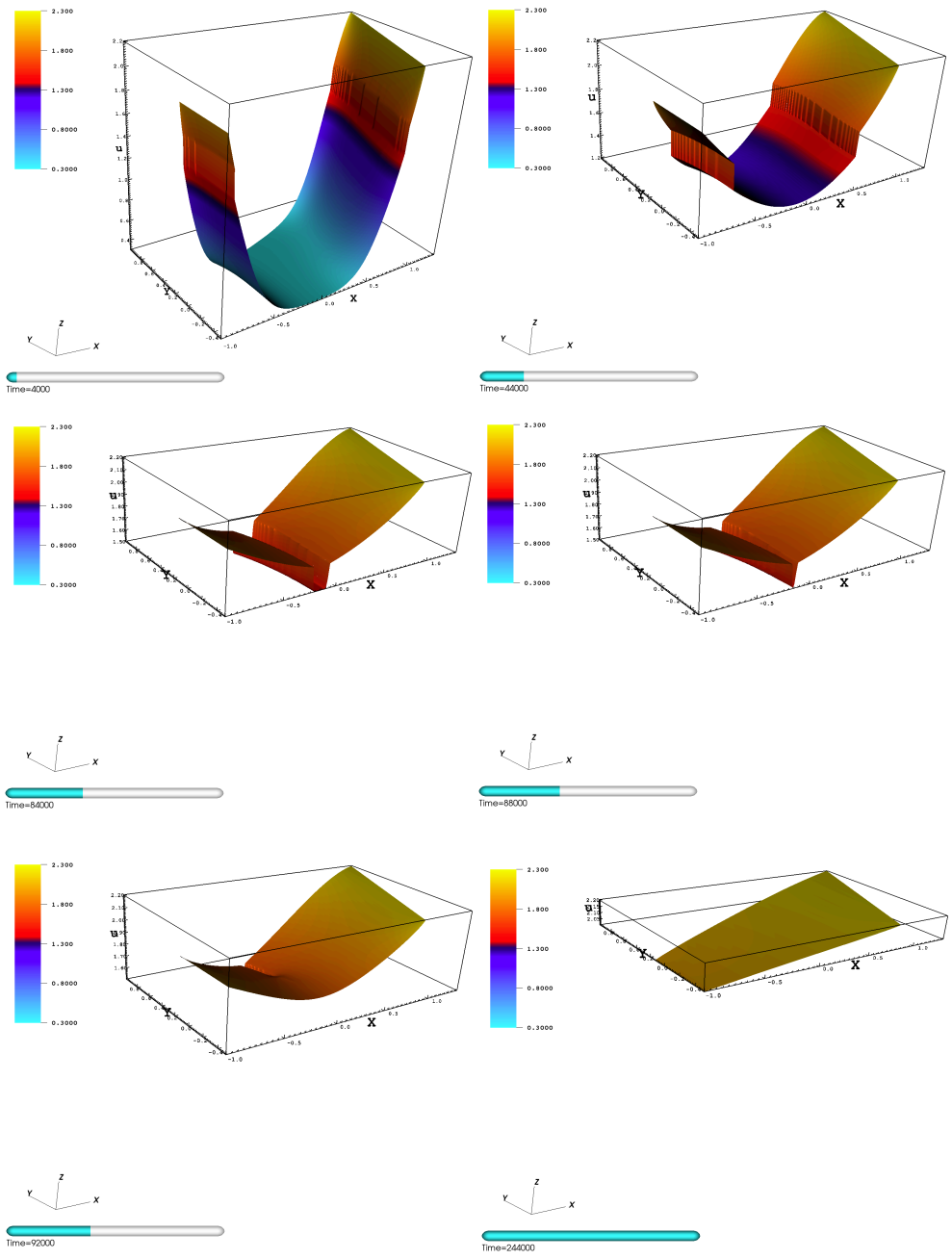


Fig. 5.20 : Specific internal energy field as a function of time, for $\mathrm{RDG}_{\mathrm{P_2P_3}}$ with ESDIRK_3. 32,768 elements. $\Delta t = 2,000$ sec ($Fo \sim 850$). Specific internal energy is scaled by $\bar{\mathrm{u}} = 4.86 \cdot 10^5 \ \frac{\mathrm{J}}{\mathrm{kg}}$.

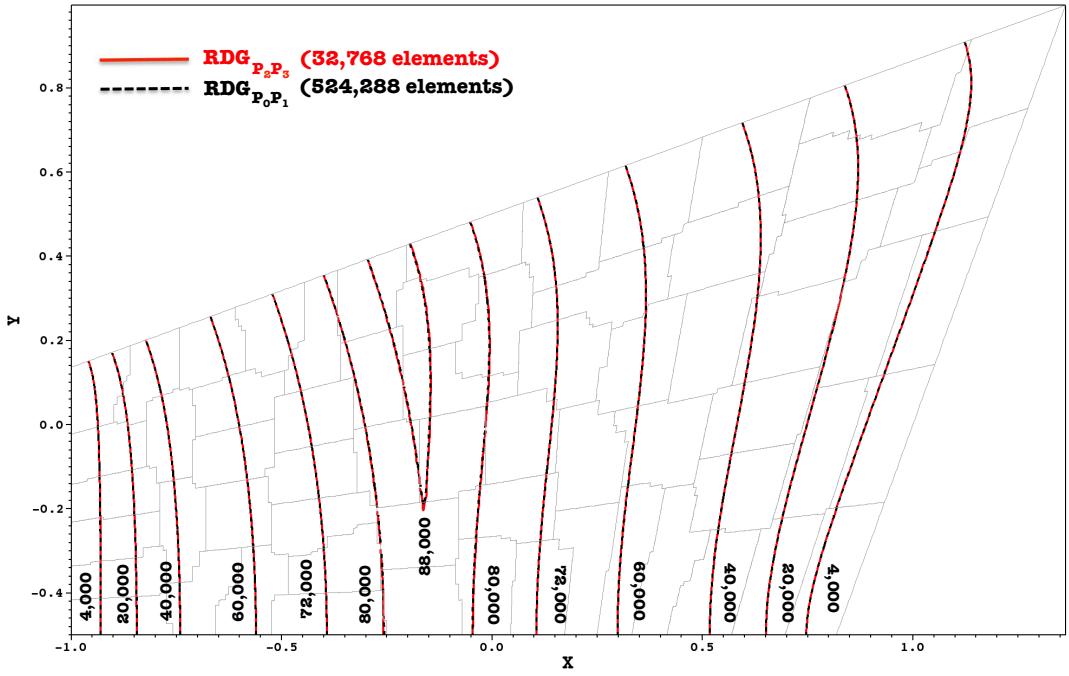


Fig. 5.21 : Comparison of $\mathrm{RDG}_{\mathrm{P_2P_3}}$ and $\mathrm{RDG}_{\mathrm{P_0P_1}}$ space discretizations, for melting front (solidus) position. Simulations with ESDIRK_3 with $\Delta t = 2,000$ sec.

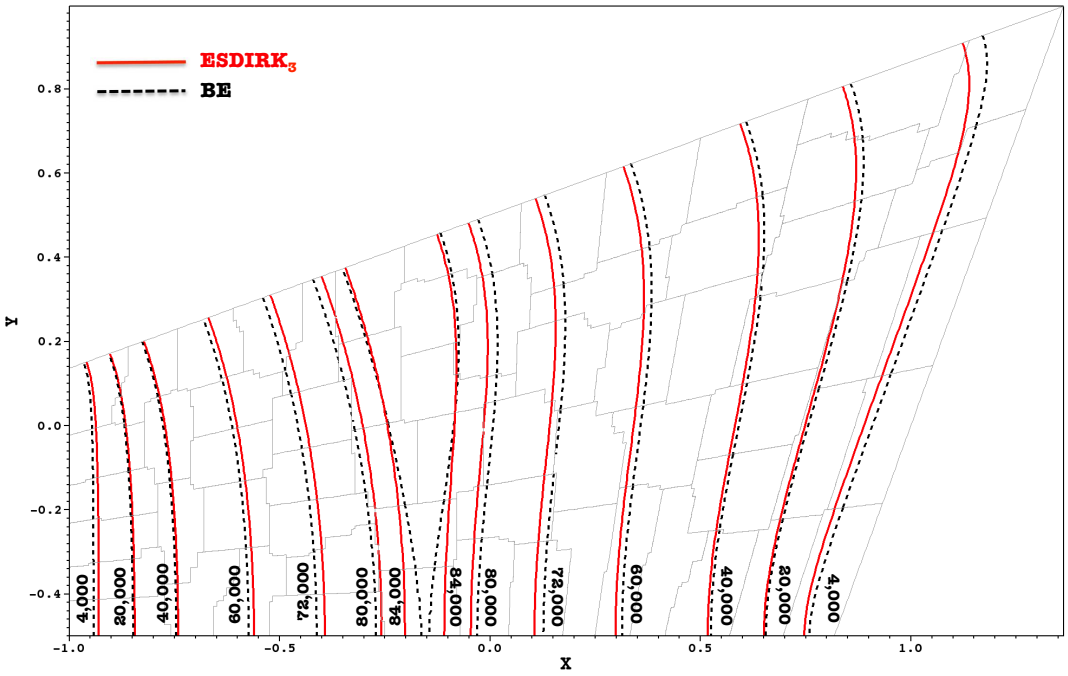


Fig. 5.22 : Comparison of ESDIRK_3 and BE time discretizations, for melting front (solidus) position. Simulation with $\mathrm{RDG}_{\mathrm{P}_2\mathrm{P}_3}$ on mesh with 32,768 elements, and $\Delta t = 2,000$ sec ($Fo \sim 850$).

performed using the $\mathrm{RDG}_{\mathrm{P_2P_3}}$ and ESDIRK_3 schemes, with resolution of 32,768 elements in total and $\Delta t = 2,000$ sec. In these figures, we also show the position of the melting front, as isolines for solidus (black solid line), and liquidus (dashed yellow line). As one can see, it takes about $25\frac{1}{2}$ hours to completely meltdown the block under the given conditions.

Next, we show the history of specific internal energy, in Figures 5.18-5.20. There is a distinctive jump in internal energy at the melting front, corresponding to latent heat. Thus, using the specific internal energy (or enthalpy) as solution variables (see Sections 2.1.2 and 2.1.3) without limiters is not recommended. This causes oscillations emanating from the melting front, and related to this problem to converge non-linear iterations, as the Newton method is having a hard time to lock onto a converging solution. On the other hand, since temperature is a smooth-across-front variable, there are no any problems to achieve robust convergent solutions, which are typically obtained within 3 to 5 non-linear iterations, using critical point secant line search algorithm [BBE+04]. It is important to note that even though non-convergent Newton solutions do not affect significantly the order of time accuracy, they do affect conservation, as the method is *fully conservative only upon convergence of non-linear iteration loop*. Here, we set non-linear tolerance to 10^{-8}.

Finally, we show the effects of discretization schemes in Figures 5.21 and 5.22. It is apparent that space discretization errors are of minor importance, as both the $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and the $\mathrm{RDG}_{\mathrm{P_2P_3}}$ result in nearly identical melting front positions, Figure 5.21. It is instructive to note that the $\mathrm{RDG}_{\mathrm{P_2P_3}}$ solution is obtained on the mesh with roughly an order of magnitude less elements, and approximately 4 times less total number of degrees of freedom (solution vector size), as compared to the $\mathrm{RDG}_{\mathrm{P_0P_1}}$ scheme solution. Due to larger element size (smaller Fo number) and smaller total solution vector size, GMRES converged faster, when using $\mathrm{RDG}_{\mathrm{P_2P_3}}$ – i.e., ~ 20 Krylov iterations per linear step in average, vs. ~ 70 Krylov iterations for $\mathrm{RDG}_{\mathrm{P_0P_1}}$. This would not be possible with non-orthogonal basis functions, as condition numbers degrade significantly with increase of the order of accuracy. In these simulations, we used an additive Schwarz (AS) combined with ILU_5 [SBG96, BBE+04] preconditioner. The smaller solution vectors and Krylov subspace sizes are advantageous also from the point of view memory, as less space required to allocate for the approximate Jacobian matrix and Krylov subspace vectors.

The effects of time discretization are more significant. As one can see from Figure 5.22, the melting front due to the first-order Backward Euler scheme is lagging behind the one predicted by the more accurate ESDIRK_3 scheme. It is worth pointing out that both schemes are unconditionally (L-) stable, allowing us to use rather aggressive time stepping of $\Delta t = 2,000$ sec. This time step corresponds to $Fo \sim 850$ for the mesh used by $\mathrm{RDG}_{\mathrm{P_2P_3}}$, and to $Fo \sim 20,000$ for the finer mesh of $\mathrm{RDG}_{\mathrm{P_0P_1}}$. These are significantly larger than stability limits of explicit schemes ($\Delta t \leq 1$), which are practically out of question for these types of applications, due to efficiency/robustness limitations.

5.5 Manufactured solution for compressible Navier-Stokes Equations

To test convergence in space and time for the problem with both hyperbolic and diffusion operators, the following solution is manufactured in 2D,

$$\begin{aligned} T(x,y) &= \bar{T} + A_{_T} \cos\left(2\pi\left(x + \mathrm{v}_{_0}\mathrm{t}\right)\right) \sin\left(2\pi\left(y + \mathrm{v}_{_1}\mathrm{t}\right)\right) \\ P(x,y) &= \bar{P} + A_{_P} \sin\left(2\pi\left(x + \mathrm{v}_{_0}\mathrm{t}\right)\right) \cos\left(2\pi\left(y + \mathrm{v}_{_1}\mathrm{t}\right)\right) \\ v_{_0}(x,y) &= A_{_v} \cos\left(2\pi\left(x + \mathrm{v}_{_0}\mathrm{t}\right)\right) \sin\left(2\pi\left(y + \mathrm{v}_{_1}\mathrm{t}\right)\right) \\ v_{_1}(x,y) &= A_{_v} \sin\left(2\pi\left(x + \mathrm{v}_{_0}\mathrm{t}\right)\right) \cos\left(2\pi\left(y + \mathrm{v}_{_1}\mathrm{t}\right)\right) \end{aligned} \tag{5.8}$$

where

$$\begin{aligned} A_{_T} &= \delta T_{_0} + a_{_T} \sin(2\pi t) \\ A_{_P} &= \delta P_{_0} + a_{_P} \sin(2\pi t) \\ A_{_v} &= \delta V_{_0} + a_{_v} \sin(2\pi t) \end{aligned} \tag{5.9}$$

and $\bar{T}$, $\bar{P}$, $\delta T_{_0}$, $\delta P_{_0}$, $\delta V_{_0}$, $a_{_T}$, $a_{_P}$, $a_{_v}$, $\mathrm{v}_{_0}$, $\mathrm{v}_{_1}$ are given constants.

Solution eq.(5.8) corresponds to translating (with velocity $\mathbf{w} = (\mathrm{v}_{_0}, \mathrm{v}_{_1})$) and oscillating (with amplitudes $a_{_T}$, $a_{_P}$ and $a_{_v}$) waves. In the following simulations, we set

$\mathbf{w}$	$=$	$\left(\frac{1}{10}, \frac{1}{10}\right)$
$\bar{P}$	$=$	1.0
$\bar{T}$	$=$	1.0
$\delta P_{_0}$	$=$	0.1
$\delta T_{_0}$	$=$	0.1
$\delta V_{_0}$	$=$	10^{-4}
$a_{_P}$	$=$	0.05
$a_{_v}$	$=$	0.01
$a_{_T}$	$=$	0.05

We used γ-gas equation of state, with $\gamma = 1.4$. Both thermal conductivity and dynamic viscosity are set to be constant, $\kappa = 0.1$ and $\mu = 0.1$. Source terms generating this manufactured solution are computed using symbolic maniputation in *Mathematica*. Computational domain and mesh were the same as in described in Section 5.2.

Dynamics of pressure and velocity magnitude fields are shown in Figures 5.23 and 5.24, obtained with high-fidelity space and time resolution.

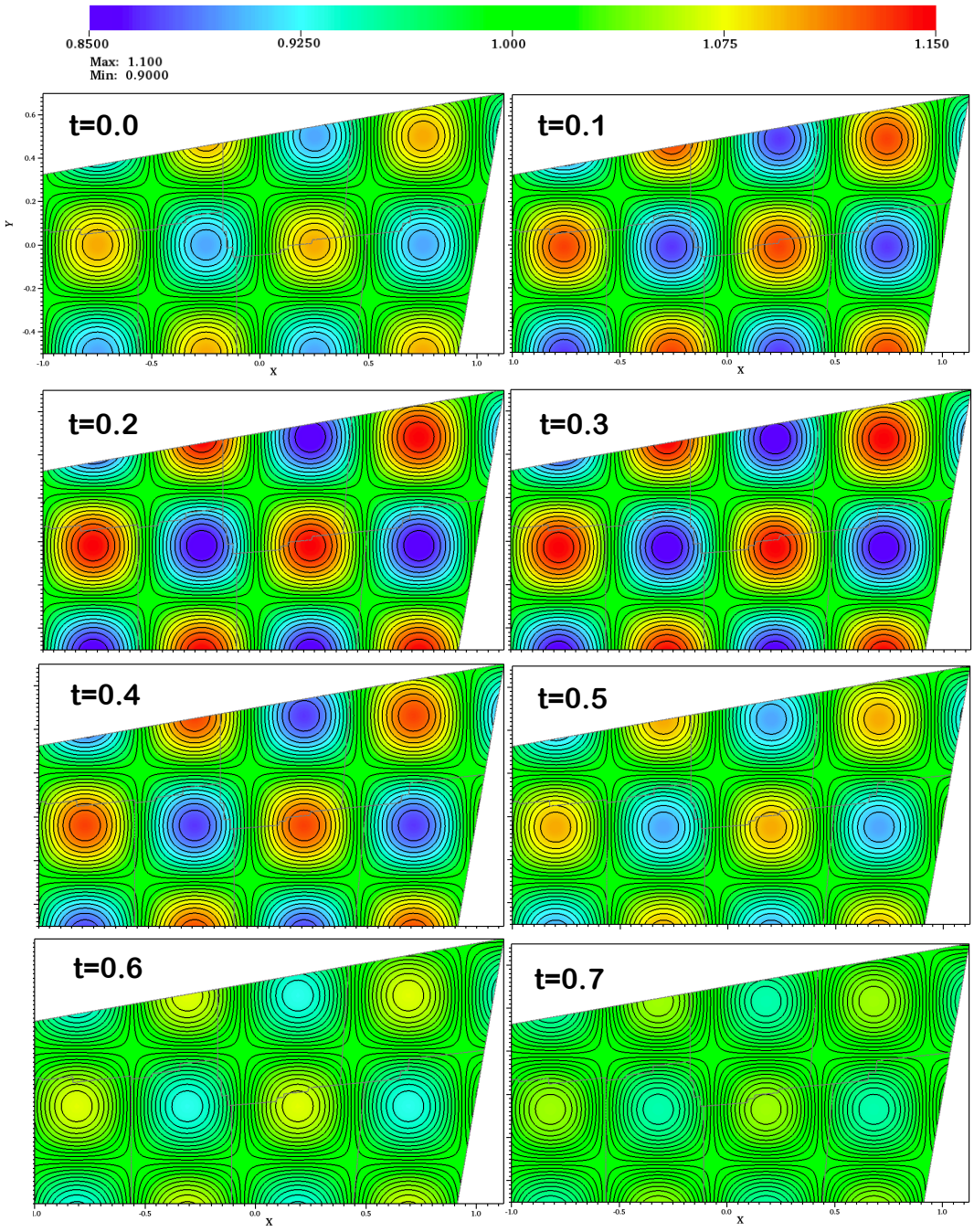


Fig. 5.23 : Dynamics of the pressure field for manufactured problem, using $\mathrm{RDG}_{\mathrm{P_2P_3}}$, ESDIRK_5, $\Delta t = 0.1$, 8,192 elements, partitioned with 7 CPUs (partitioning is also shown).

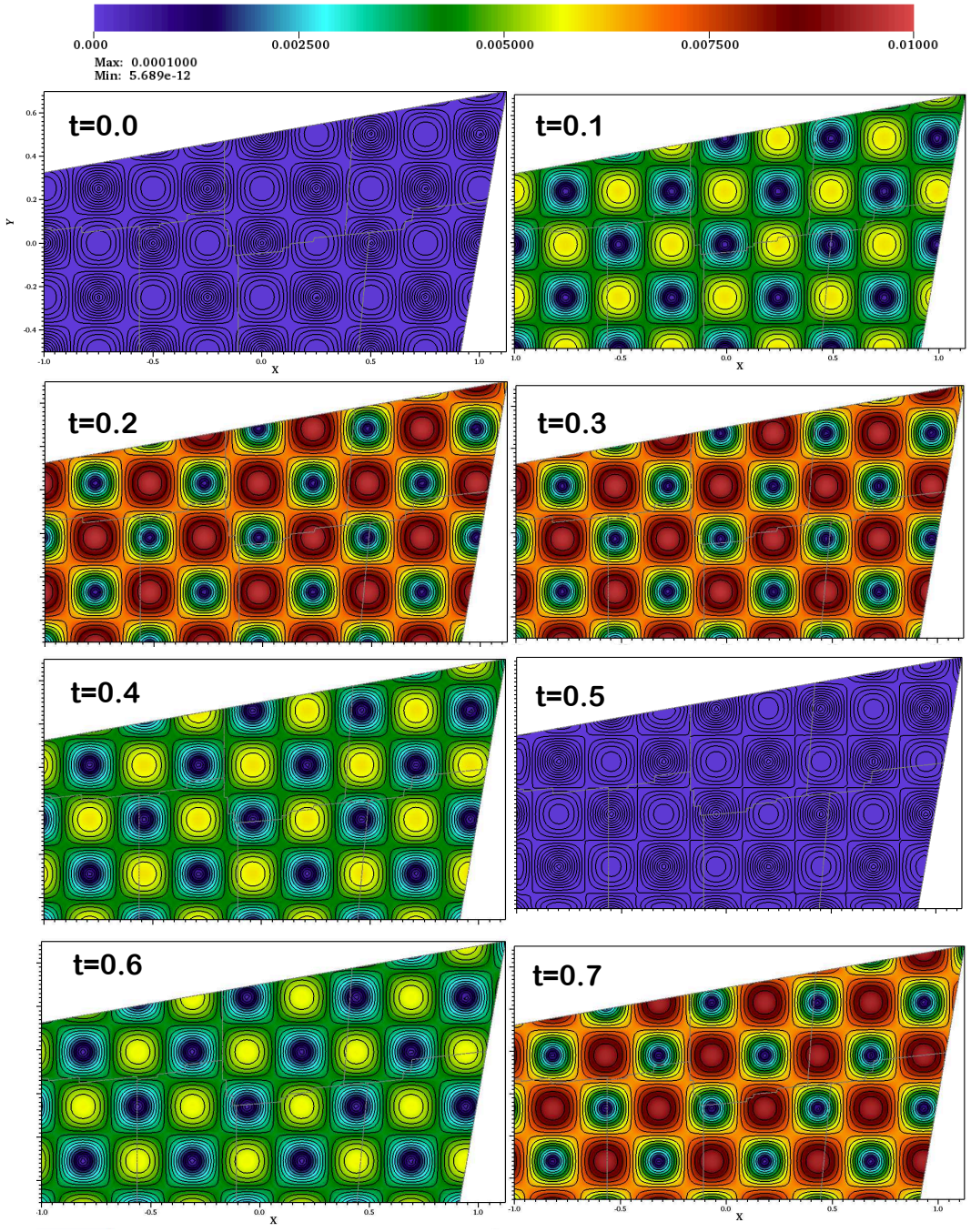


Fig. 5.24 : Dynamics of the velocity magnitude field for manufactured problem, using $\mathrm{RDG}_{\mathrm{P_2P_3}}$, ESDIRK_5, $\Delta t = 0.1$, 32,762 elements, partitioned with 7 CPUs (partitioning is also shown).

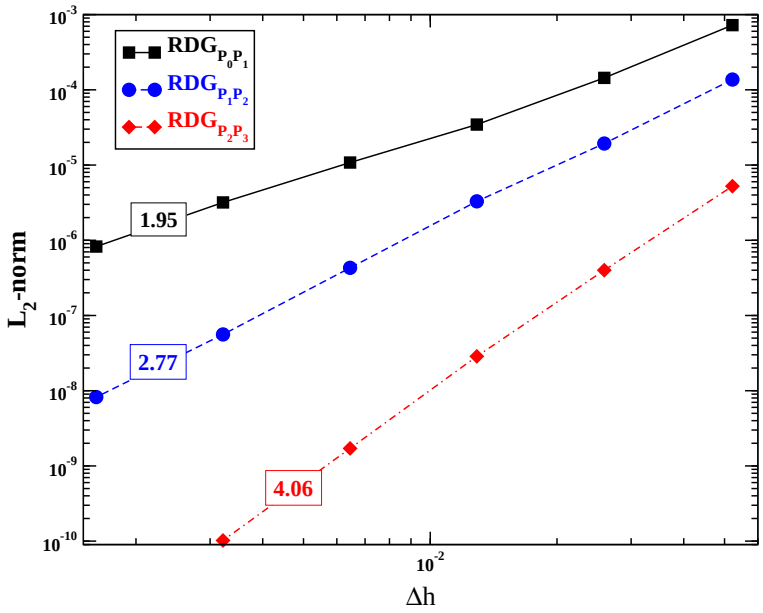


Fig. 5.25 : On convergence of the pressure field, with mesh refinement and different space discretization schemes.

First, we measure space convergence rates, by using the 5^{th}-order-accurate time discretization ESDIRK$_5$ and setting time step to $2 \times \Delta t = 0.001$. This ensures that time discretization errors are smaller than space discretization errors. The results are shown in Figures 5.25 and 5.26, for pressure and velocity magnitude convergence, respectively. As one can see, all three RDG schemes do converge consistently – i.e., with the second order for $\mathrm{RDG}_{\mathrm{P_0P_1}}$, with the third-order for $\mathrm{RDG}_{\mathrm{P_1P_2}}$, and with (approximately) the fourth-order for $\mathrm{RDG}_{\mathrm{P_2P_3}}$.

Time convergence is demonstrated in Figures 5.27 and 5.28, for pressure and velocity magnitude, respectively. In the simulations, we used $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh with 32,762 elements, to ensure small spatial discretization errors. This space resolution is sufficient to measure nearly asymptotic convergence rates for time discretization schemes up to the 3^{rd} order accurate. The fourth- and the fifth-order accurate ESDIRK$_{4,5}$ schemes exhibit nearly 4^{th} order convergence rate, when time steps are large. The convergence is flatten for smaller time steps, when space discretization errors become dominant.

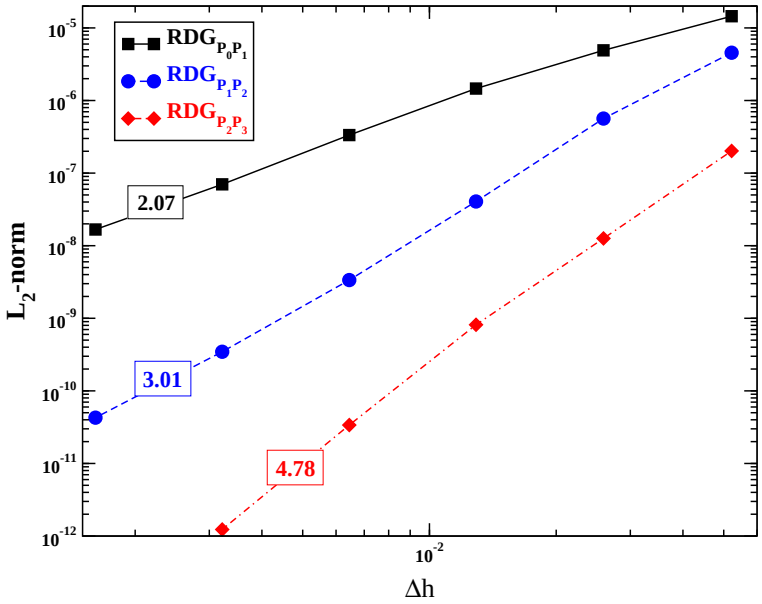


Fig. 5.26 : On convergence of the velocity field, with mesh refinement and different space discretization schemes.

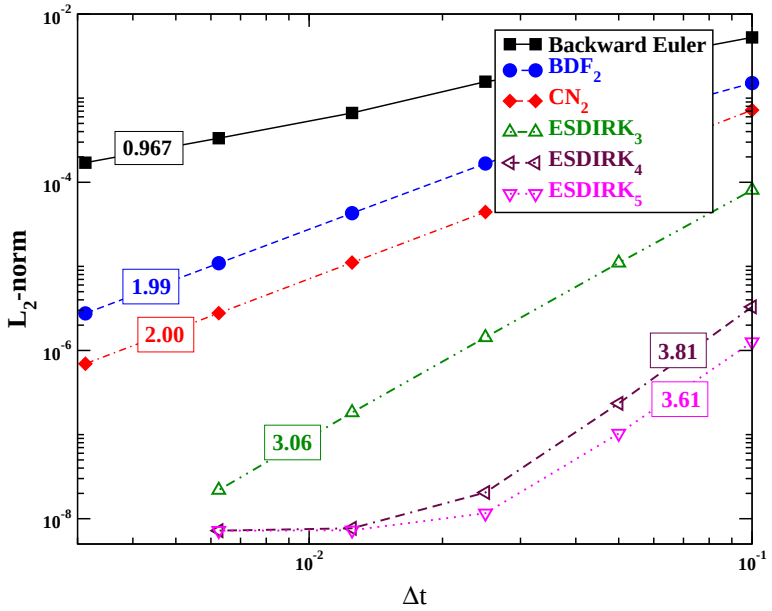


Fig. 5.27 : On convergence of the pressure field, with time step refinement and different time discretization schemes.

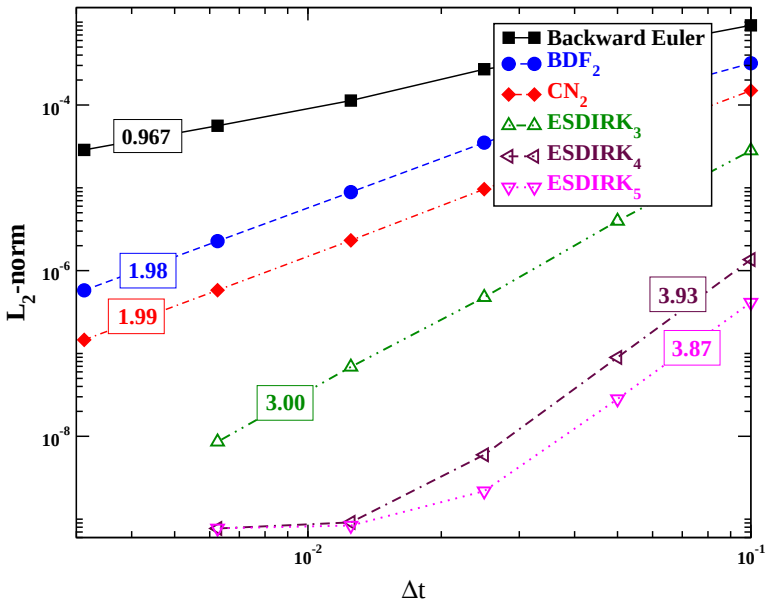


Fig. 5.28 : On convergence of the velocity field, with time step refinement and different time discretization schemes.

5.6 Lid-Driven Cavity

In the present section, we will test no-slip boundary conditions for compressible Navier-Stokes solver, using the "Lid-Driven Cavity" ("LDC") flow. This problem has been established as a standard "benchmark" test for numerical methods of (nearly-) incompressible fluid dynamics. In [GGS82], Ghia et al. employed finite-difference method, using steady-state vorticity-stream-function ($\omega - \psi$) formulation of incompressible flow, to obtain solutions in a wide range of Re numbers. Using grids with high resolution (129×129 and 257×257) and "coupled strongly implicit multigrid" method, Ghia et al. provided "reference" data for comparison of velocity profiles at the vertical and horizontal centerlines of the square cavity, with no-slip vertical and bottom walls, and moving top wall.

Here, we will simulate this problem using compressible formulation, setting flow Ma number to low values. We will use γ-gas law, with $\gamma = 1.4$. Initially, the gas is motionless, under $\rho = 1$ and pressure set to

$$P = \frac{1}{\gamma M_0^2}$$

where M_0 is flow Mach based on lid velocity. At $t = 0$, the lid velocity is set to 1. Fluid temperature is set to 1, with specific heat defined as

$$C_v = \frac{1}{\gamma\left(\gamma - 1\right) M_0^2}$$

Wall temperatures are kept under $T_w = 1$. In most simulations, we use $M_0 = 10^{-2}$. We used second-order BDF_2 time discretization, starting with time step $\Delta t = 10^{-4}$, and rapidly increasing it to $\Delta t = 1$ with factor 2.

It is instructive to note that (material/acoustic) CFL numbers with this time step are approximately (32/32,000), (64/64,000) and (128/12,800) for mesh resolutions 32×32, 64×64 and 128×128, correspondingly. While it is evident that this time step is sufficient to resolve dynamic time scales of the problem (evolution and break-up of eddies), these would be impossible time steppings for explicit and operator-splitting time discretization schemes.

5.6.1 Re=400 results

We start with low-Re number, $Re = 400$, which exhibits steady-state solution reached at $t = 30$. There are two secondary vortices formed in the lower corners

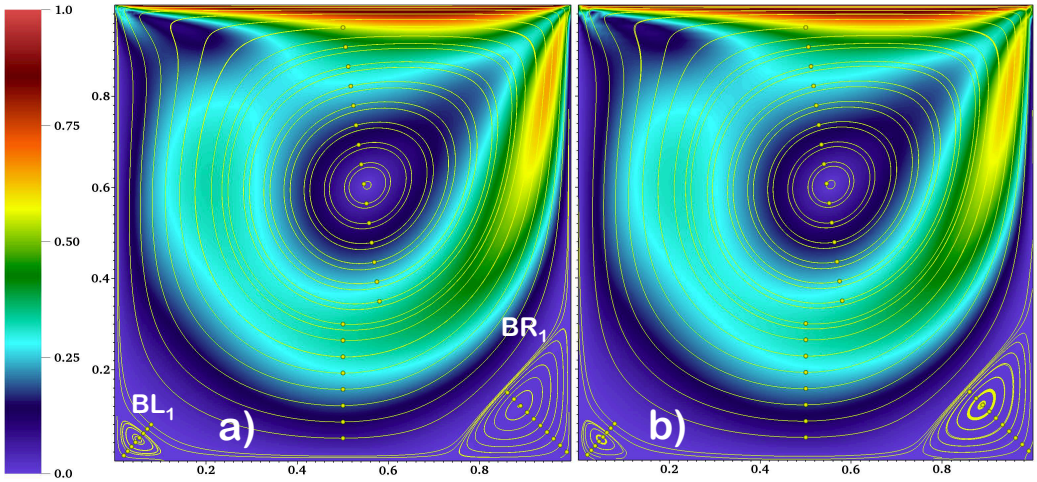


Fig. 5.29 : Velocity magnitude and streamlines, for solutions with a) $\mathrm{RDG}_{\mathrm{P_0P_1}}$, mesh 128×128, and b) $\mathrm{RDG}_{\mathrm{P_1P_2}}$, mesh 64×64.

of the cavity (BR_1 and BL_1, following notation from [GGS82]). All three tested space discretization schemes ($\mathrm{RDG}_{\mathrm{P_0P_1}}$, $\mathrm{RDG}_{\mathrm{P_1P_2}}$ and $\mathrm{RDG}_{\mathrm{P_2P_3}}$) have no problems to resolve these, as shown in Figure 5.29 for $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and $\mathrm{RDG}_{\mathrm{P_1P_2}}$. It is instructive to note that we visualized high-order solution by mapping high-order solutions to nodal values, which in turn are post-processed (by VisIt) piecewise-linearly. This does degrade quite significantly visible solutions[2]. A few markers are placed in certain positions, to enable visualization of streamlines.

Figures 5.30 and 5.31 present the calculated velocity profiles at the cavity's vertical and horizontal centerlines. From these plots, one can clearly see that high-order solutions are much more accurate than both the second-order $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and vorticity-stream-function solution from [GGS82] – as the solutions on the mesh 32×32 are rendering plots almost indistiguishable from low-order schemes, obtained with mesh 128×128.

[2] A better "multi-resolution" option has been recently introduced in VisIt. We discuss this in Section 5.7.

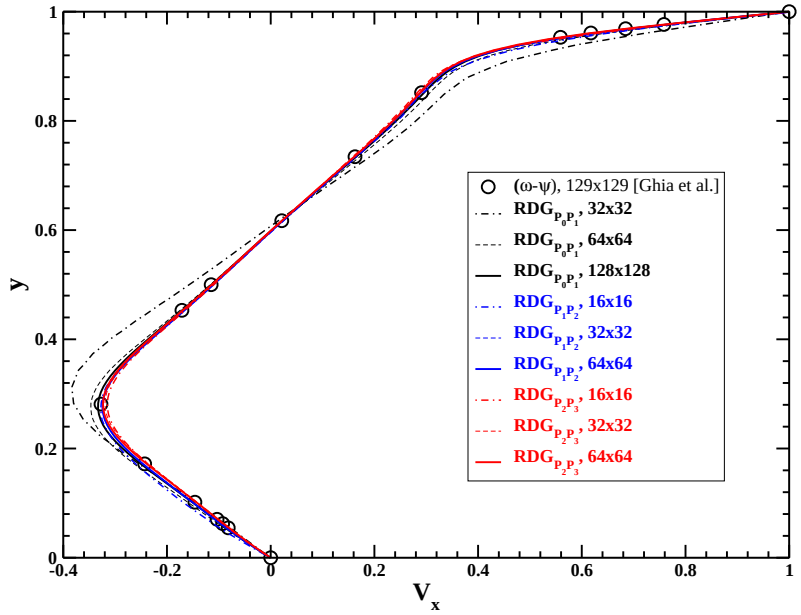


Fig. 5.30 : Comparison of the velocity profile at the cavity's verical centerline, for $Re = 400$.

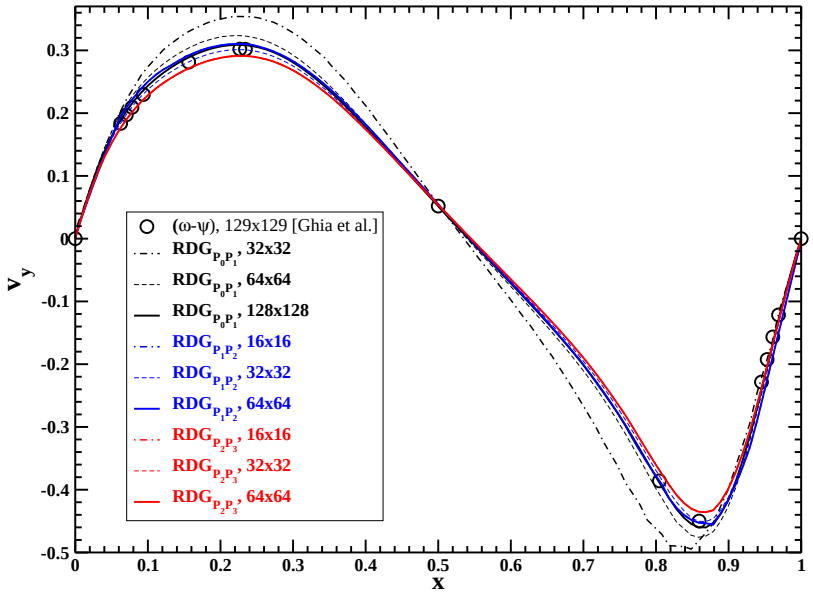


Fig. 5.31 : Comparison of the velocity profile at the cavity's horizontal centerline, for $Re = 400$.

5.6.2 Re=10,000 results

Next, we move to more difficult case with high-Re-number $Re = 10^4$. This is the case when advantages of high-order schemes become more evident.

First, we observe that the solution is not steady-state, as under these high-Reynolds-number conditions the eddies are constantly evolving: forming, breaking, dissappearing. This can be seen from Figures 5.32 and 5.33, showing a sequence of snapshots for velocity magnitude and streamlines, obtained with the fourth-order-accurate $\mathrm{RDG}_{\mathrm{P_2P_3}}$ scheme on mesh 64×64. This makes direct comparison to [GGS82] difficult, as Ghia et al. assumed steady flow solution. Nevertheless, we observed numerous secondary vortices formed, $\mathrm{BR}_{1,2}$ and $\mathrm{BL}_{1,2}$, which are dynamically growing and decaying. Also, the large central vortex is unstable and gyrating with a period of approximately 10 time units.

Grid convergence is shown in Figure 5.36, for two space discretizations - $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and $\mathrm{RDG}_{\mathrm{P_2P_3}}$, using a sequence of meshes with resolution ranging from 16×16 to 256×256. It is evident that the second-order scheme $\mathrm{RDG}_{\mathrm{P_0P_1}}$ is significantly more diffusive than the fourth-order-accurate $\mathrm{RDG}_{\mathrm{P_2P_3}}$ – as the top wall "boundary layer" and the "jet" forming the primary vortex are thicker under the coarser mesh. With a better mesh, the solution with the $\mathrm{RDG}_{\mathrm{P_0P_1}}$ tends to approach to the one for the $\mathrm{RDG}_{\mathrm{P_2P_3}}$. However, even with the 256×256 mesh, the $\mathrm{RDG}_{\mathrm{P_0P_1}}$ is still significantly more diffusive than the $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh 64×64. This is also evident from the higher level of dumping of the gyrating motion for the primary vortex, as one can see from Figures 5.33 vs. 5.35 (the "gyration" offset is significantly smaller[3]). In order to get a comparable second-order solution, one should probably solve for using the mesh of 512×512, which corresponds to the case of more than ten times more total degrees of freedom. This, in addition to 64 times larger CFL, makes all related underlying linear algebra much more stiff. Combining with more memory requirements for storage of solution (Krylov) vectors and (approximate Jacobian) matrices for preconditioning – the higher-order solutions are much more cost-effective, for these types of flow. Advantages of the higher-order RDG become even more significant in three dimensions.

Figure 5.37 demostrates an ability to resolve subcell flow structures, using

[3] Under the coarser meshes of 64×64 and lower, this gyrating motion is completely dumped by the $\mathrm{RDG}_{\mathrm{P_0P_1}}$.

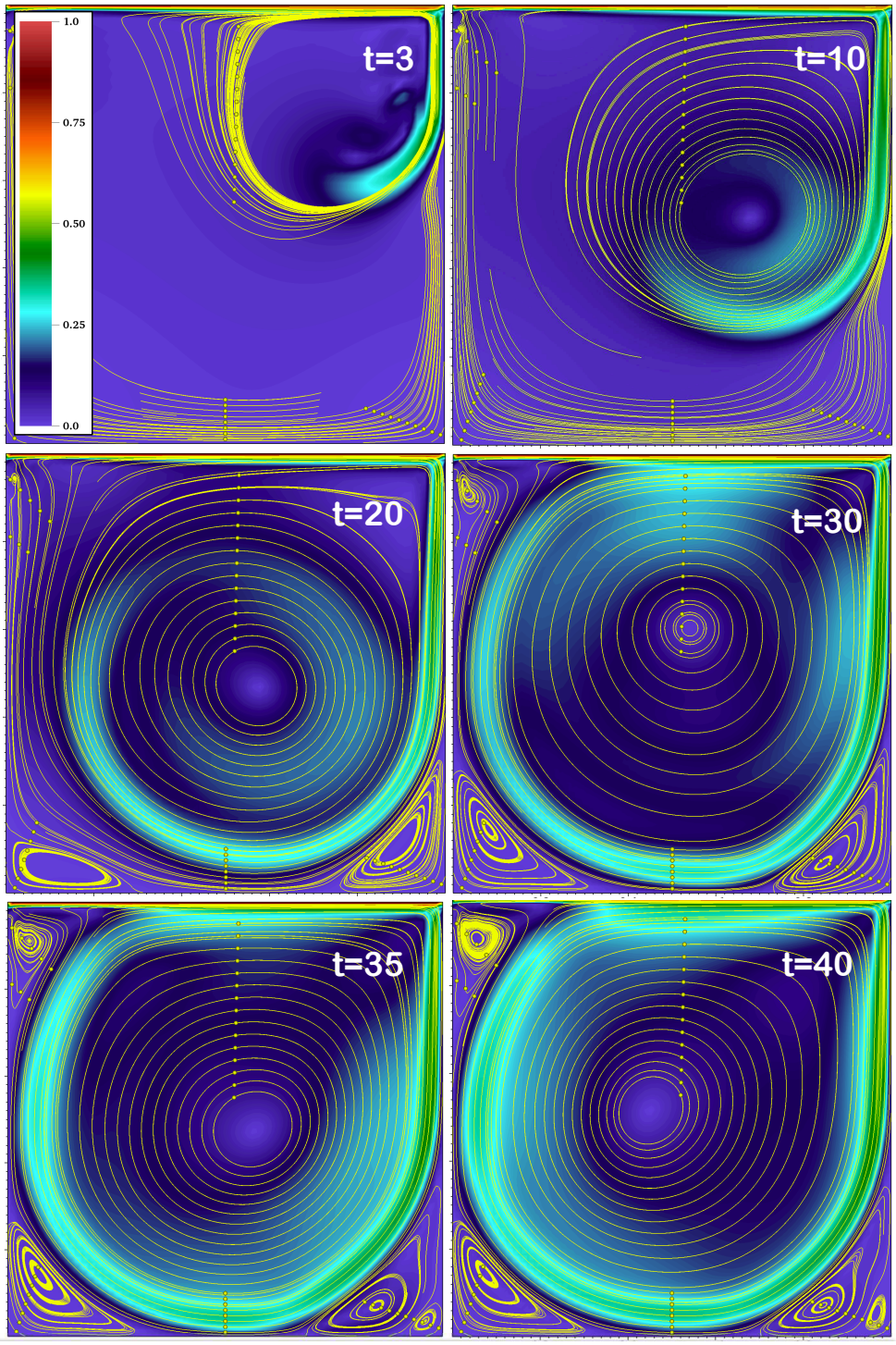


Fig. 5.32 : Dynamics of the velocity magnitude and streamlines for $Re = 10^4$, $\mathrm{RDG}_{\mathrm{P_2P_3}}$ and mesh resolution 64×64.

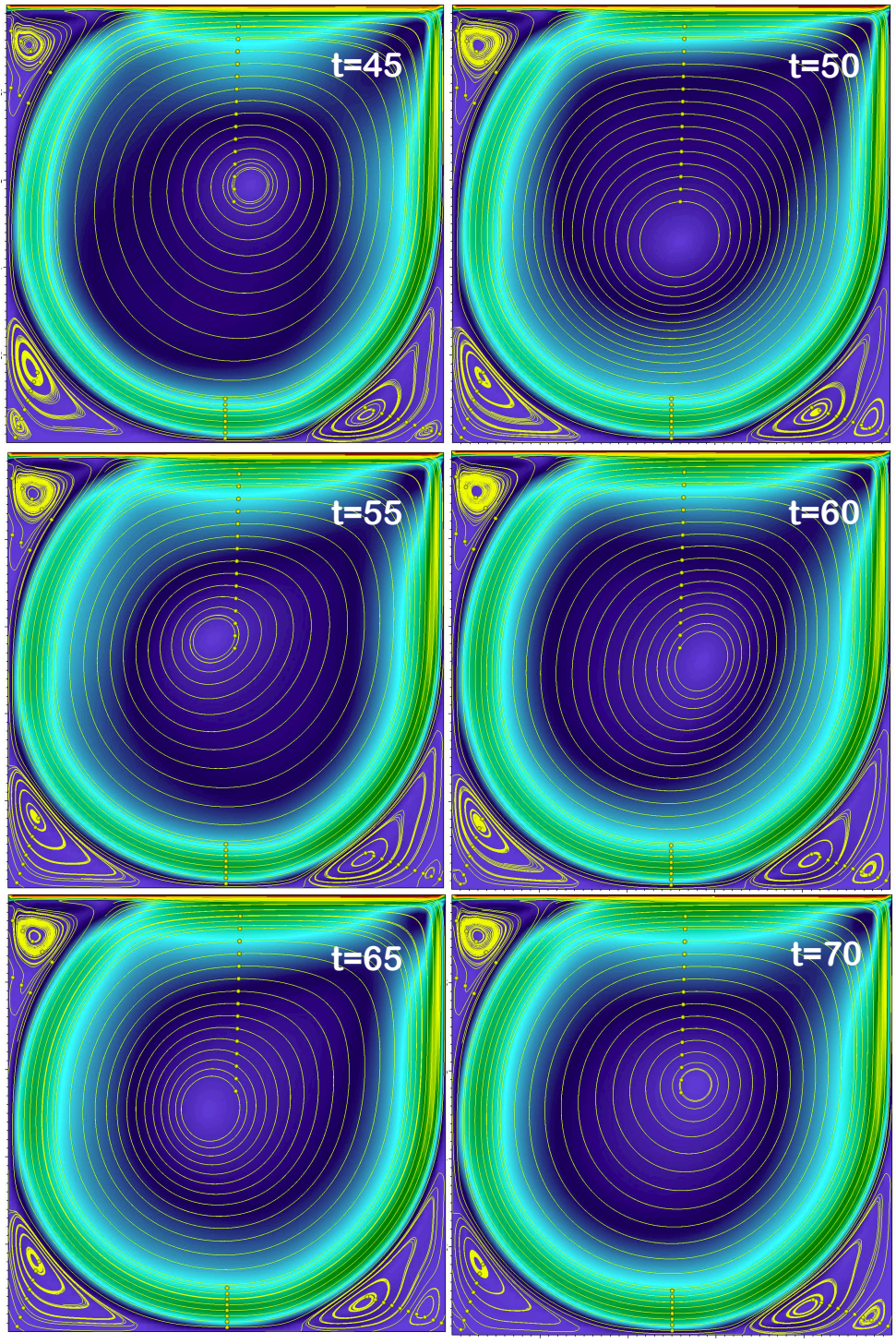


Fig. 5.33 : Dynamics of the velocity magnitude and streamlines for $Re = 10^4$, $\mathrm{RDG}_{\mathrm{P_2P_3}}$ and mesh resolution 64×64 (continued).

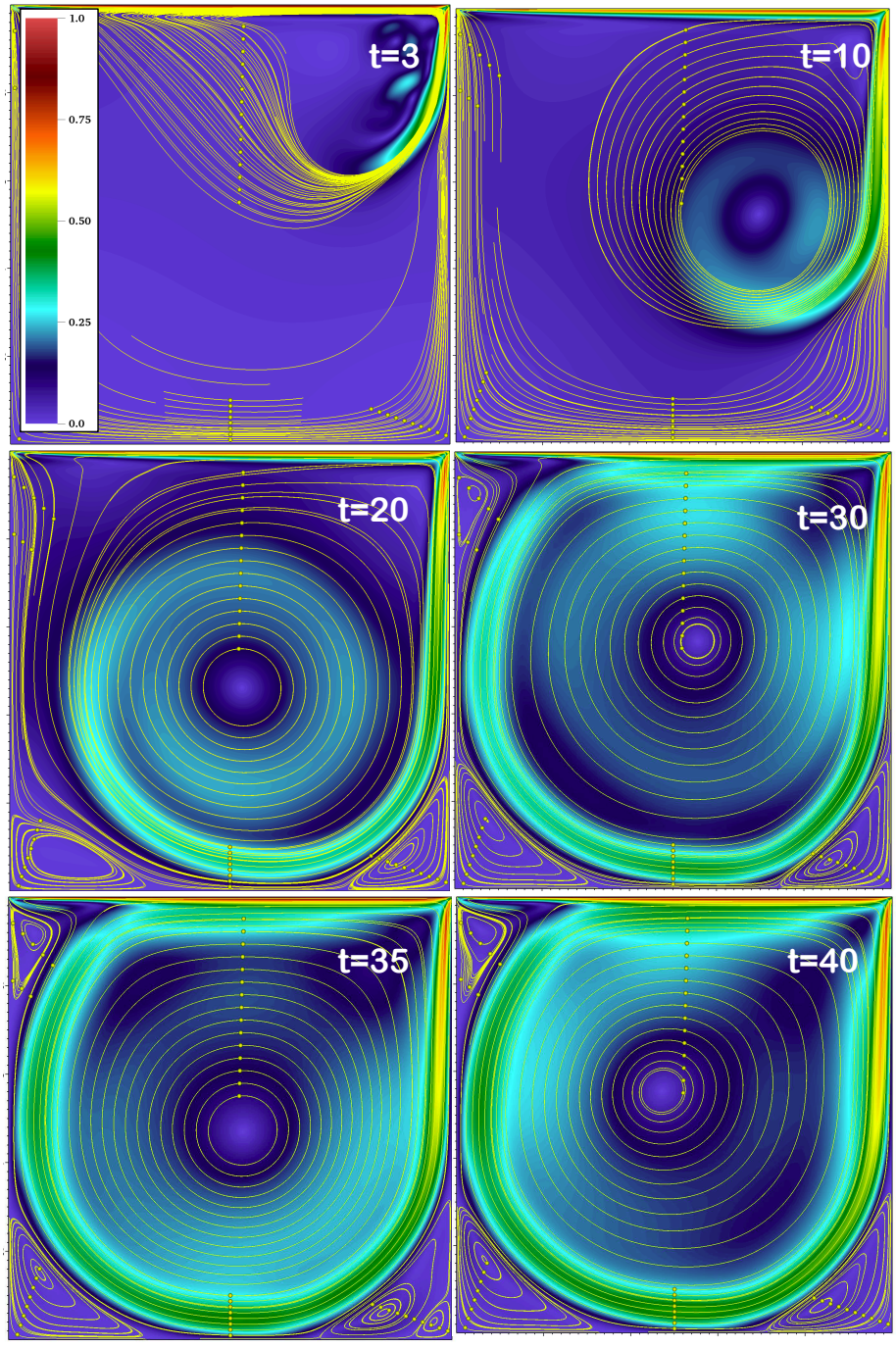


Fig. 5.34 : Dynamics of the velocity magnitude and streamlines for $Re = 10^4$, $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and mesh resolution 256×256.

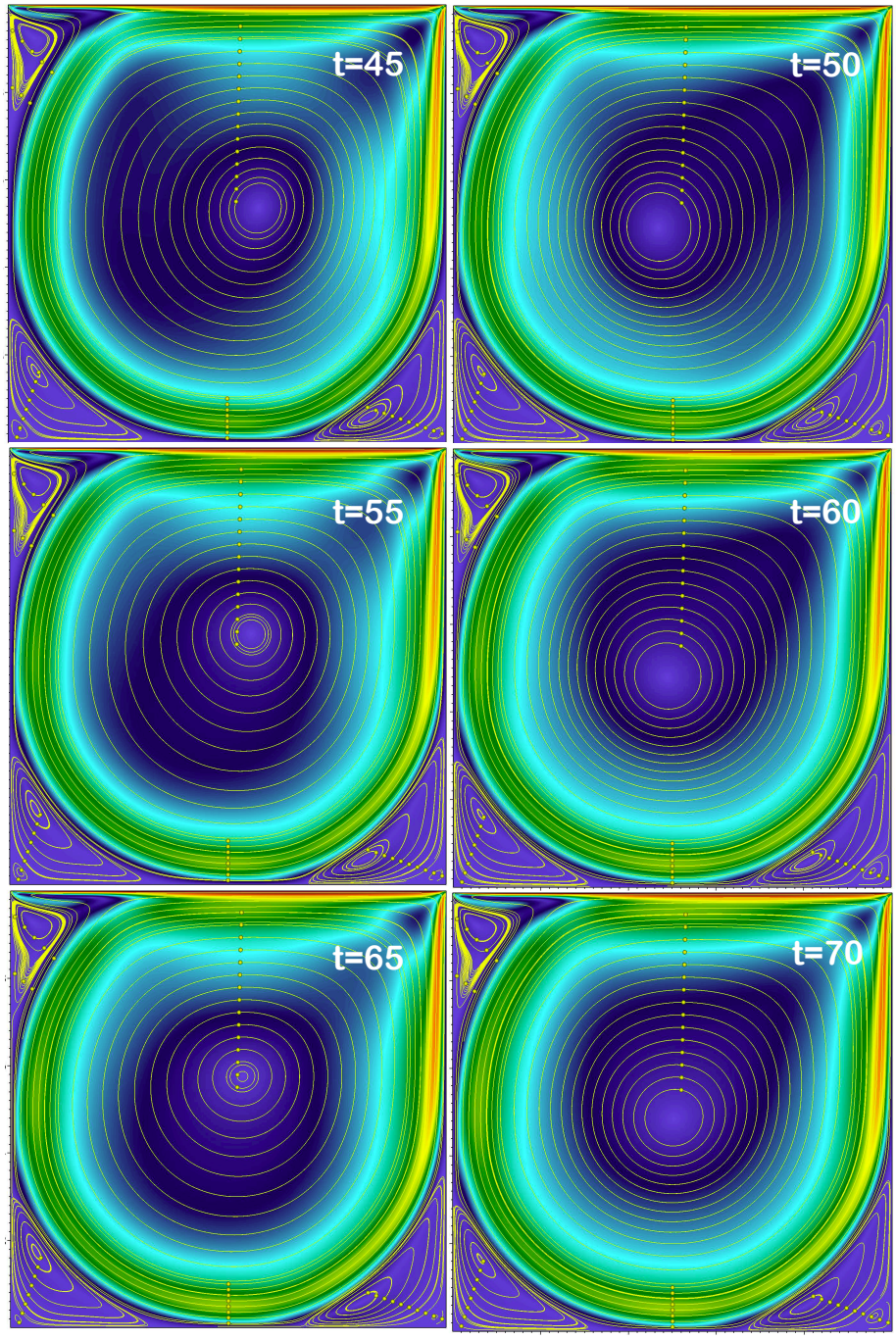


Fig. 5.35 : Dynamics of the velocity magnitude and streamlines for $Re = 10^4$, $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and mesh resolution 256×256 (continued).

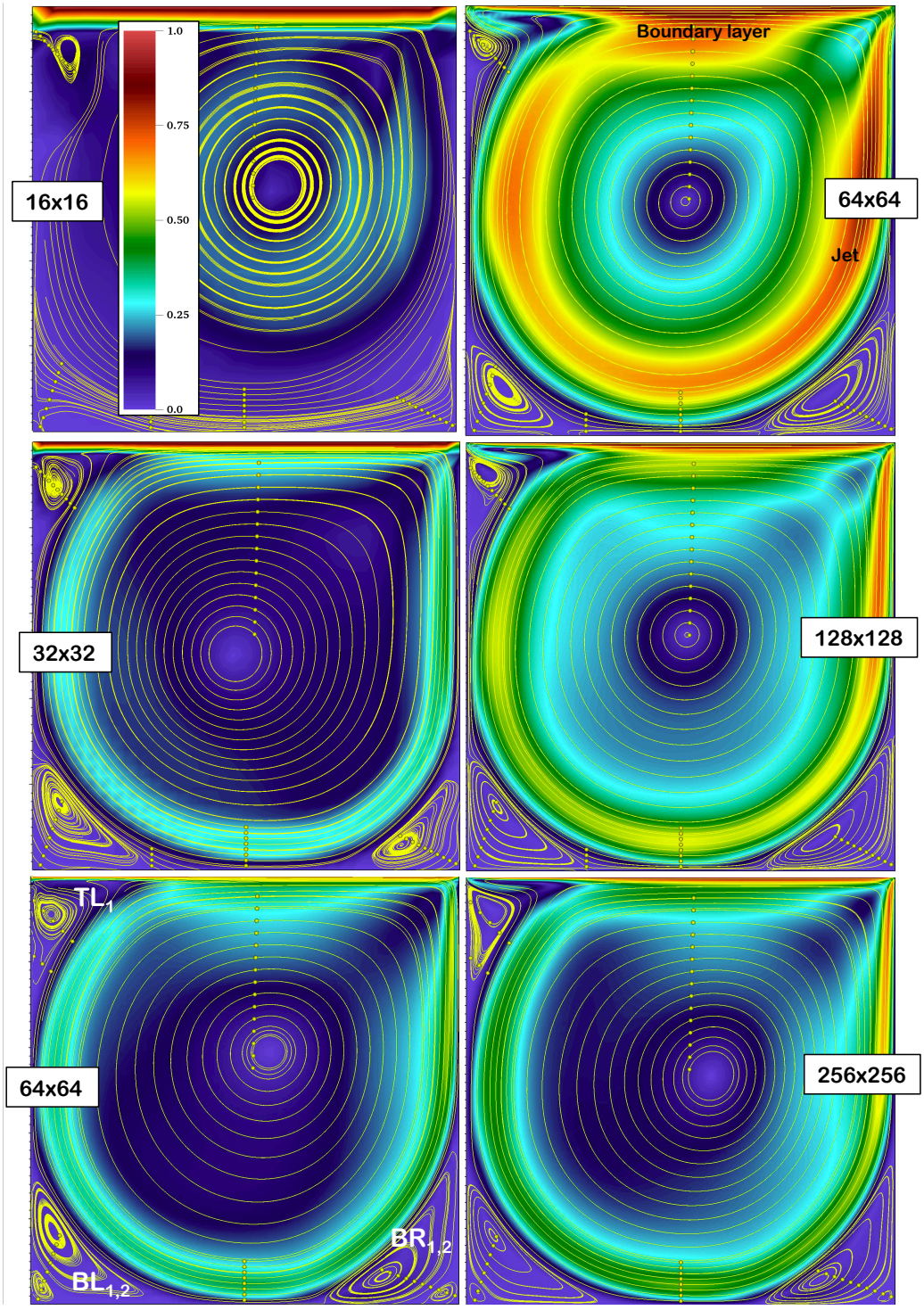


Fig. 5.36 : On grid convergence for $\mathrm{RDG}_{\mathrm{P_0P_1}}$ (right) and $\mathrm{RDG}_{\mathrm{P_2P_3}}$ (left), for $Re = 10^4$, $t = 45$.

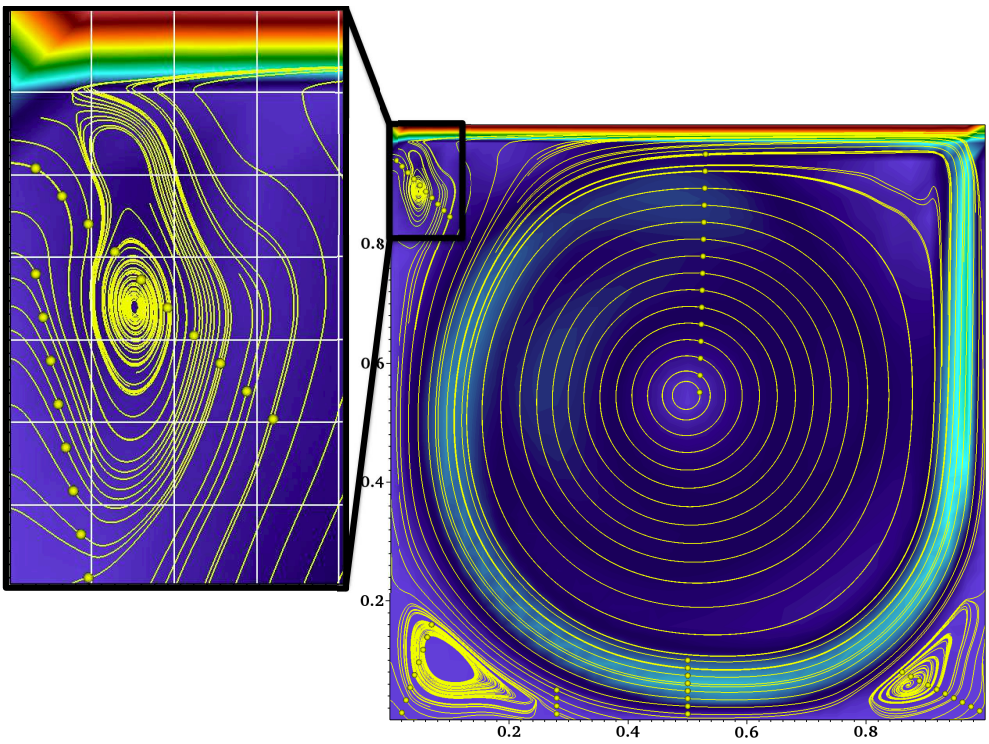


Fig. 5.37 : On resolution of the TL_1 vortex and top wall boundary layer with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ and mesh 32×32. $Re = 10^4$, $t = 28$.

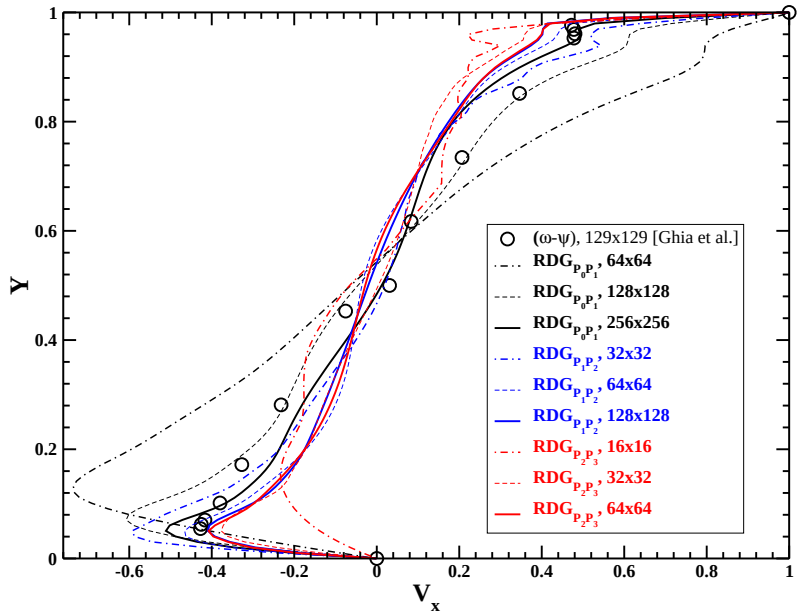


Fig. 5.38 : Comparison of the velocity profile at the cavity's verical centerline, for $Re = 10^4$, $t = 70$.

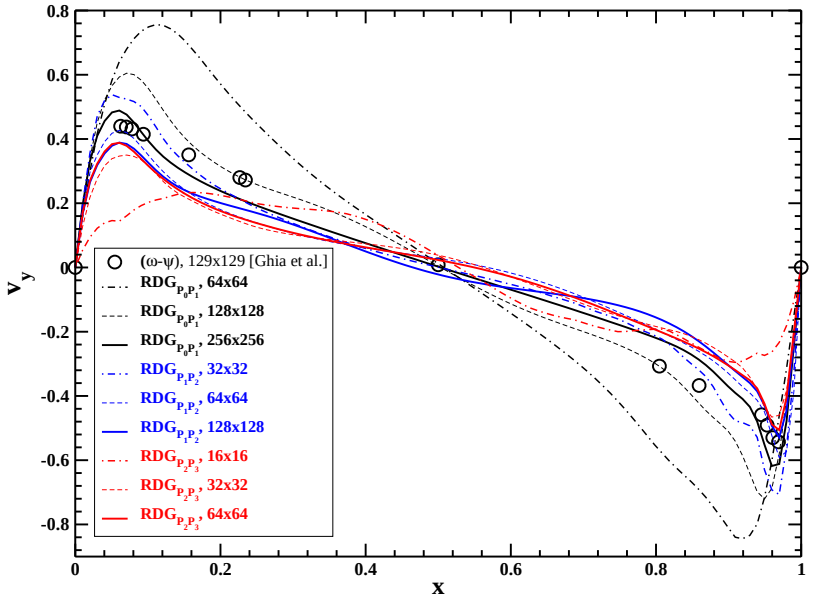


Fig. 5.39 : Comparison of the velocity profile at the cavity's horizontal centerline, for $Re = 10^4$, $t = 70$.

the 4^{th} order piecewise-cubic $\mathrm{RDG}_{\mathrm{P_2P_3}}$ scheme. It is evident that with this low 32×32 mesh, all secondary vortices are of the grid size scale. Also, the entire top wall boundary layer is completely subcell. Nevertheless, the solver is able to capture all the major flow features with decent accuracy. The second-order $\mathrm{RDG}_{\mathrm{P_0P_1}}$ scheme on the same mesh failed to capture TL_1 vortex. Also, the top wall boundary layer is severely smeared out.

It is instructive to note that the flow field depicted in Figure 5.37 is actually post-processed by the visualization tool (VisIt), converting nodal velocity representation (originally piecewise-cubic) to piecewise-linear fields. The actual piecewise-cubic solution is better.

Finally, we show comparison of instantaneous velocity profiles at vertical and horizontal centerlines, in Figures 5.38 and 5.39. Because of the transient nature of the flow, direct comparison with steady-state solution by [GGS82] is difficult. Nevertheless, from these plots, we can clearly see the importance of space resolution/order of accuracy for these high-Reynolds-number dynamically-evolving vortical flows. Notice that solutions with $\mathrm{RDG}_{\mathrm{P_1P_2}}$ on mesh 128×128 and $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh 64×64 are nearly identical, which is an indicator of sufficient spatial resolution. On the other hand, while $\mathrm{RDG}_{\mathrm{P_0P_1}}$ profiles do tend to approach $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh 64×64, with better mesh refinement, they are still not there. It would require mesh resolution in exceed of 512×512, in order to get a comparable solution. This would be significantly more expensive than $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh 64×64 – both in solution vector size (and memory requirements), and CPU time.

5.6.3 Aspect ratio 2, Re=10,000 results

Next, we performed simulations of driven-cavity flows using a box with aspect ratio 2. All simulations are done using the fourth-order $\mathrm{RDG}_{\mathrm{P_2P_3}}$ and mesh resolution 32×64, varying Mach number from 10^{-2} to 0.3. The results are shown in Figures 5.40 and 5.41. Similar to the square cavity case, the flow is unsteady. There are several secondary vortices dynamically formed in the lower half of the cavity, as shown in Figure 5.40. There are some differences in the flow dynamics under different Mach numbers, mostly associated with different timing for secondary vortical structures formation and break-up, Figure 5.41. It is instructive to note that CPU time for these three distinct Mach numbers is approximately the same.

We are able to push these compressible flow simulations to smaller Mach numbers. However, below a certain threshold (in this case, $M \leq 10^{-3}$), the round-off errors become significant, preventing a clean solution using γ-gas equation of state. Note, that in order to reduce Mach number (to increase the speed of sound), one should increase initial pressure as

$$P \sim \frac{\rho c^2}{\gamma}$$

Thus, for $M = 10^{-3}$, initial pressure is on the order of 10^6. In this case, it is better to switch to incompressible flow formulation, which enforces filtering undesirable acoustic modes and related inability to solve for non-linear problem tightly enough, to completely eliminate round-off errors.

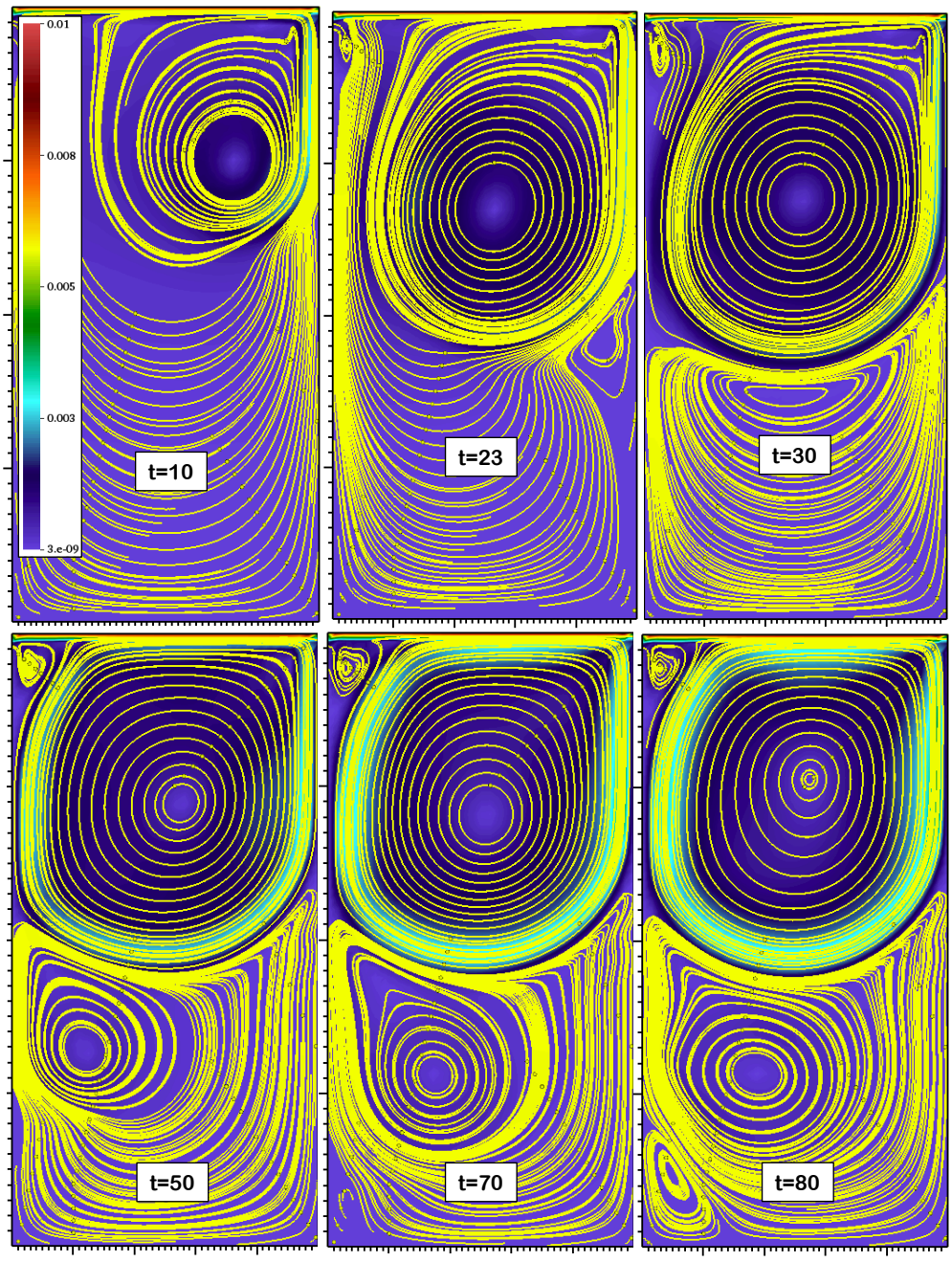


Fig. 5.40 : Dynamics of Mach number field and streamlines, using $\mathrm{RDG}_{\mathrm{P_2P_3}}$ and mesh resolution 32×64, for $M = 10^{-2}$.

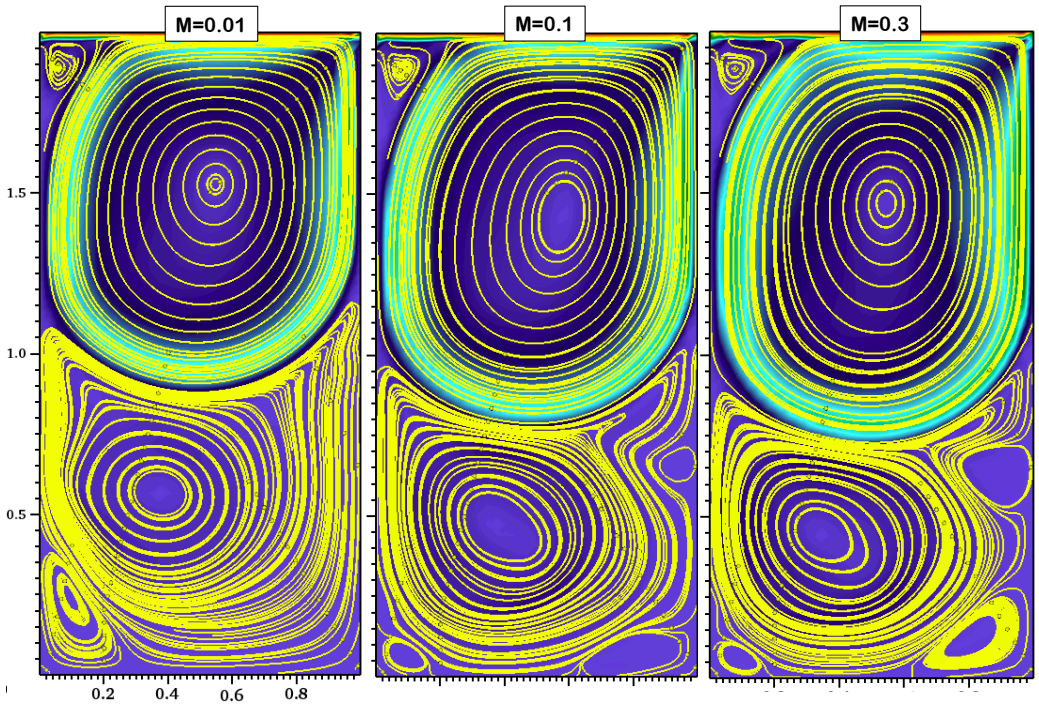


Fig. 5.41 : Comparison of the solution for Mach number field and streamlines, at $t = 80$, with different Mach numbers.

5.7 Shedding Flow past a Triangular Wedge

In the next test, we investigate performance of the orthogonal RDG for resolution of vortices in the vortex shedding flow behind a triangular wedge. The two-dimensional wedge is an equilateral triangle, with the side length $d = 1$. It is placed in the computational domain of size 40×30, five units downstream of the inlet. We used inflow boundary conditions at the left, outflow (Neumann) boundary conditions at the right, and free-stream boundary conditions at the top and bottom, Figure 5.42. At the wedge surface, no-slip isothermal boundary conditions are enforced.

Three successively refined meshes were generated, designed to provide fine resolution of boundary layers near the wedge. The meshes are depicted in Figure 5.42, and denoted as R1, R2 and R4 – with (2,055), (8,220) and (32,880) QUAD4 elements, correspondingly. Notice that the mesh has high-aspect-ratio elements in the boundary layer. Also, there are transition zones with high ratio of element sizes. We intentionally use these (generally considered "bad") meshes – as they are known to be difficult for many discretization schemes.

For time discretization, we used BDF_2, starting with $\Delta t = 10^{-3}$, increasing it rapidly to $\Delta t = 1$, with time step increase ratio 2. With this time step, the material Courant number was around 8.4, 16.8 and 33.7, for mesh resolutions R1, R2 and R4, correspondingly. Acoustic Courant numbers were approximately 840, 1680 and 3370, respectively. On the finest mesh R4, the viscous Fourier number was around 10.

The Reynolds number (based on the wedge side length) is set to $Re = 100$ ($\nu = 10^{-2}$). With proper dimensionalization, inlet/freestream conditions are $U_0 = 1$, $T_0 = 1$. We use γ-gas law, with $\gamma = 1.4$. Initial pressure was set to $P_0 = 7142.857143$, which corresponds to $M_0 = 10^{-2}$ and $\rho_0 = 1$, based on the freestream conditions. By setting $C_v = 1.7857 \cdot 10^4$ and $\kappa = 357$, the Prandtl number is 0.7. Thus, the dimensionless temperature is around 1. At the wedge surface, we set isothermal boundary conditions, with $T_w = 1$. Initial conditions are $T = 1$, $\mathbf{v} = 0$ and $P = P_0$.

Under these flow conditions, the flow behind the wedge forms a Karman street. The unsteady vortices in the Karman street are resolved with proper space-time resolution. The shedding (if any) starts at dimensionless time (see Section 2.2.7

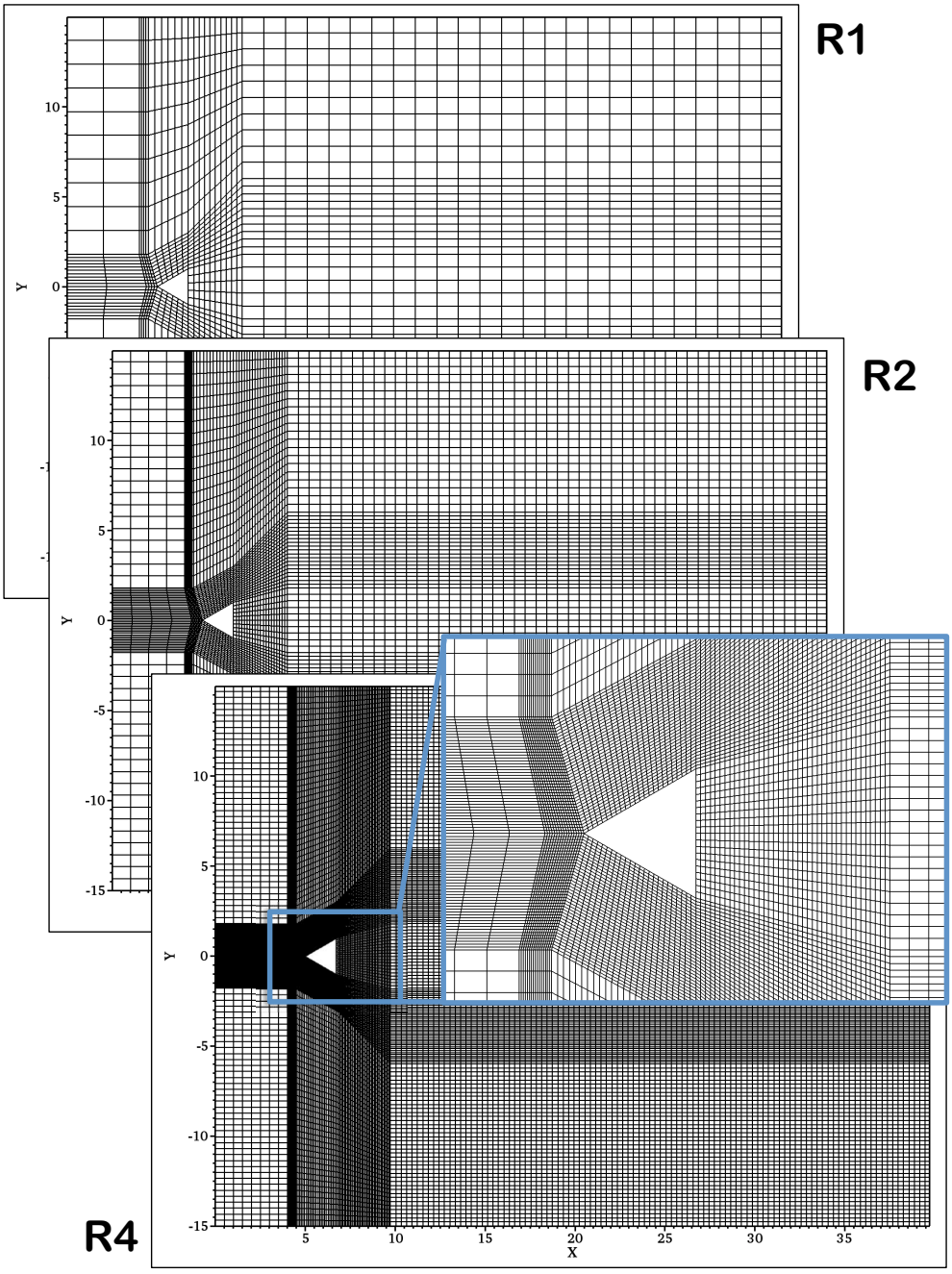


Fig. 5.42 : Meshes utilized for shedding flow past a triangular wedge.

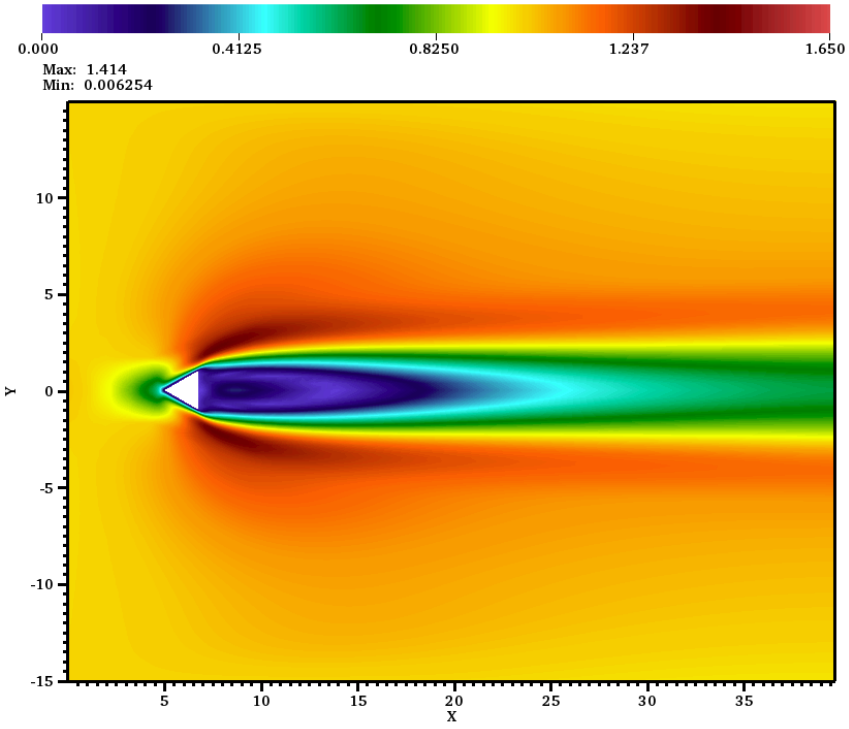


Fig. 5.43 : Steady wake behind the wedge, with $\mathrm{RDG}_{\mathrm{P_0P_1}}$ on mesh R2. Magnitude of the velocity vector, at $t = 200$.

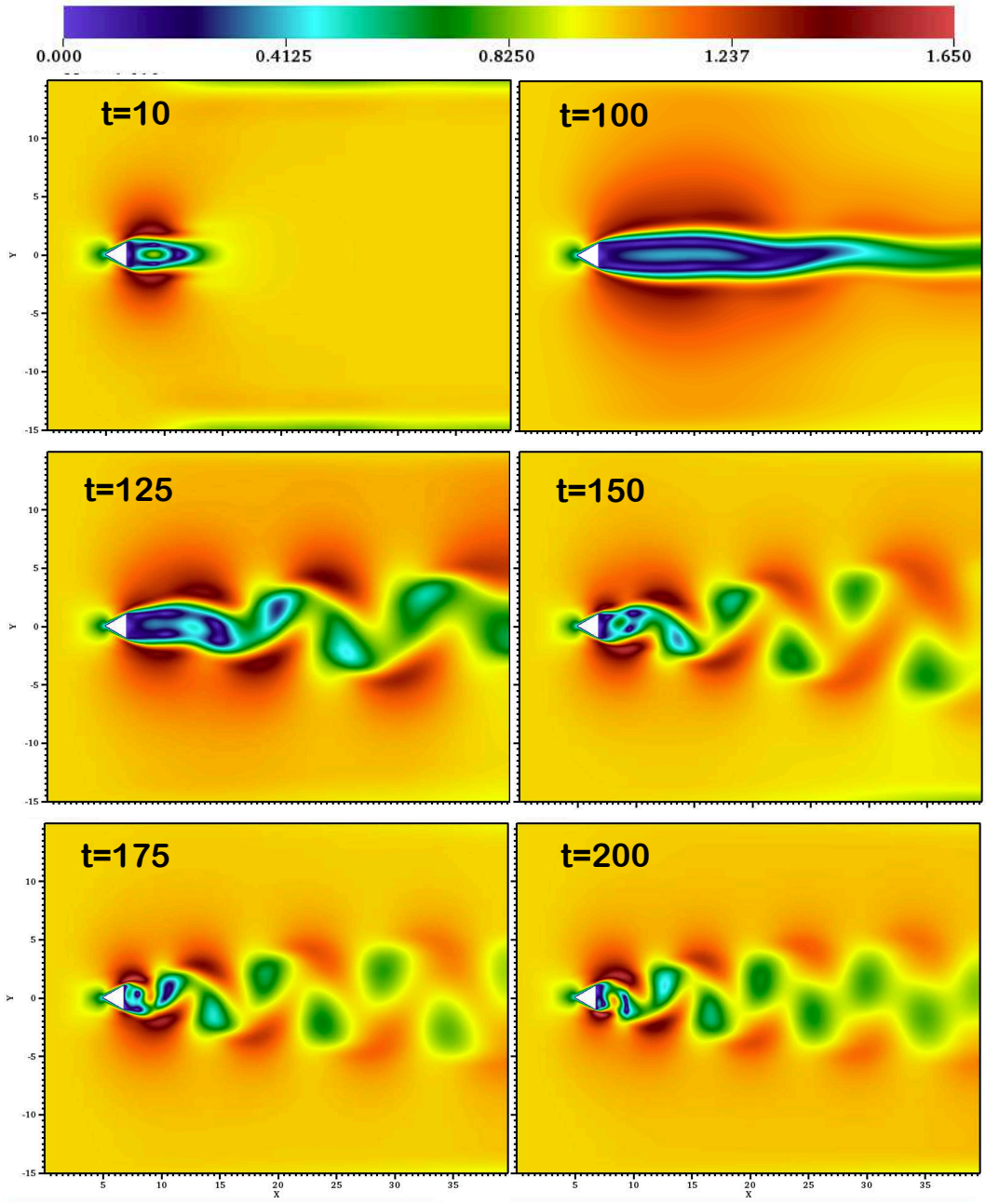


Fig. 5.44 : Vortex shedding behind the wedge, with $\mathrm{RDG}_{\mathrm{P_0P_1}}$ on mesh R4. Magnitude of the velocity vector.

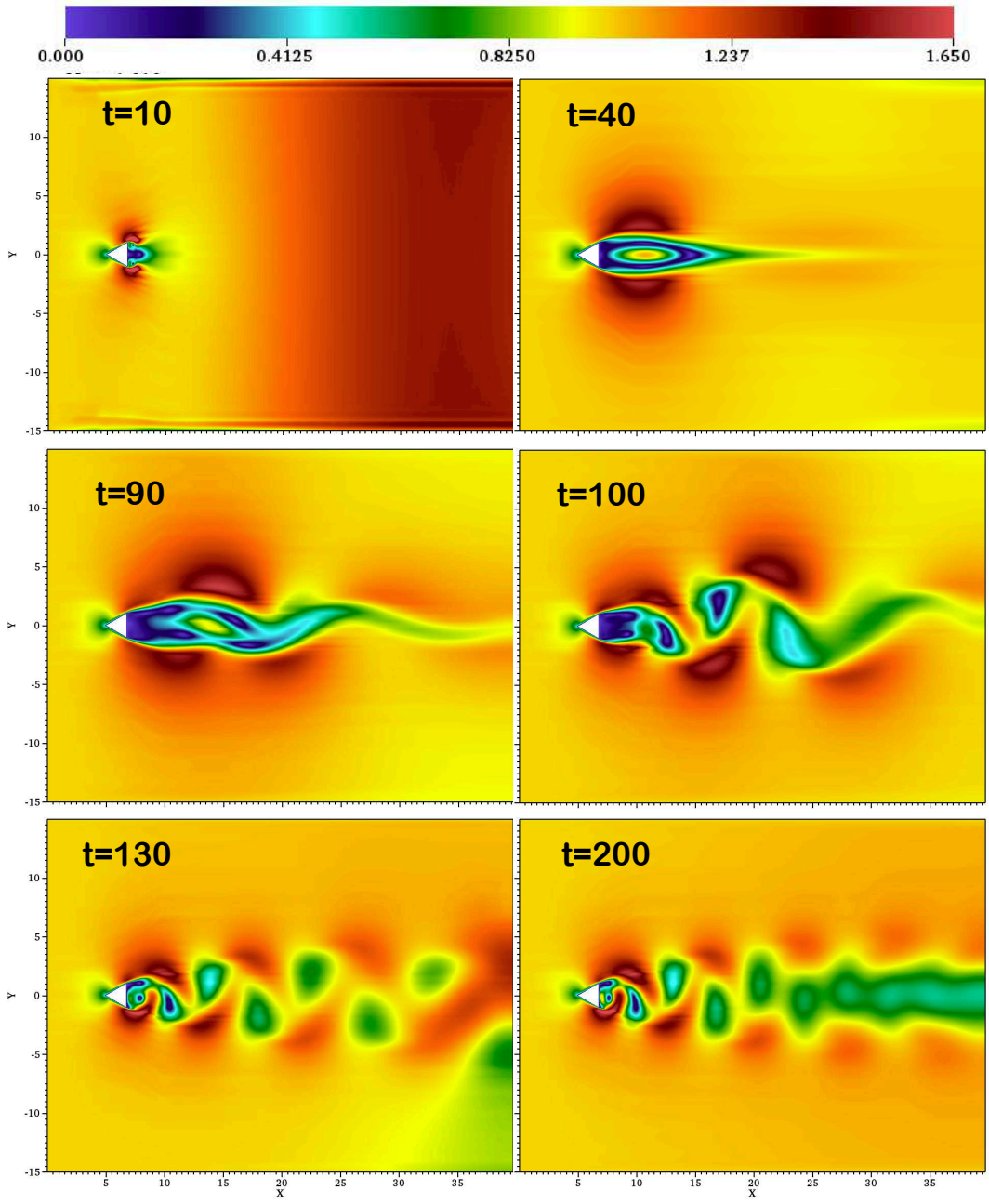


Fig. 5.45 : Vortex shedding behind the wedge, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh R1. Magnitude of the velocity vector.

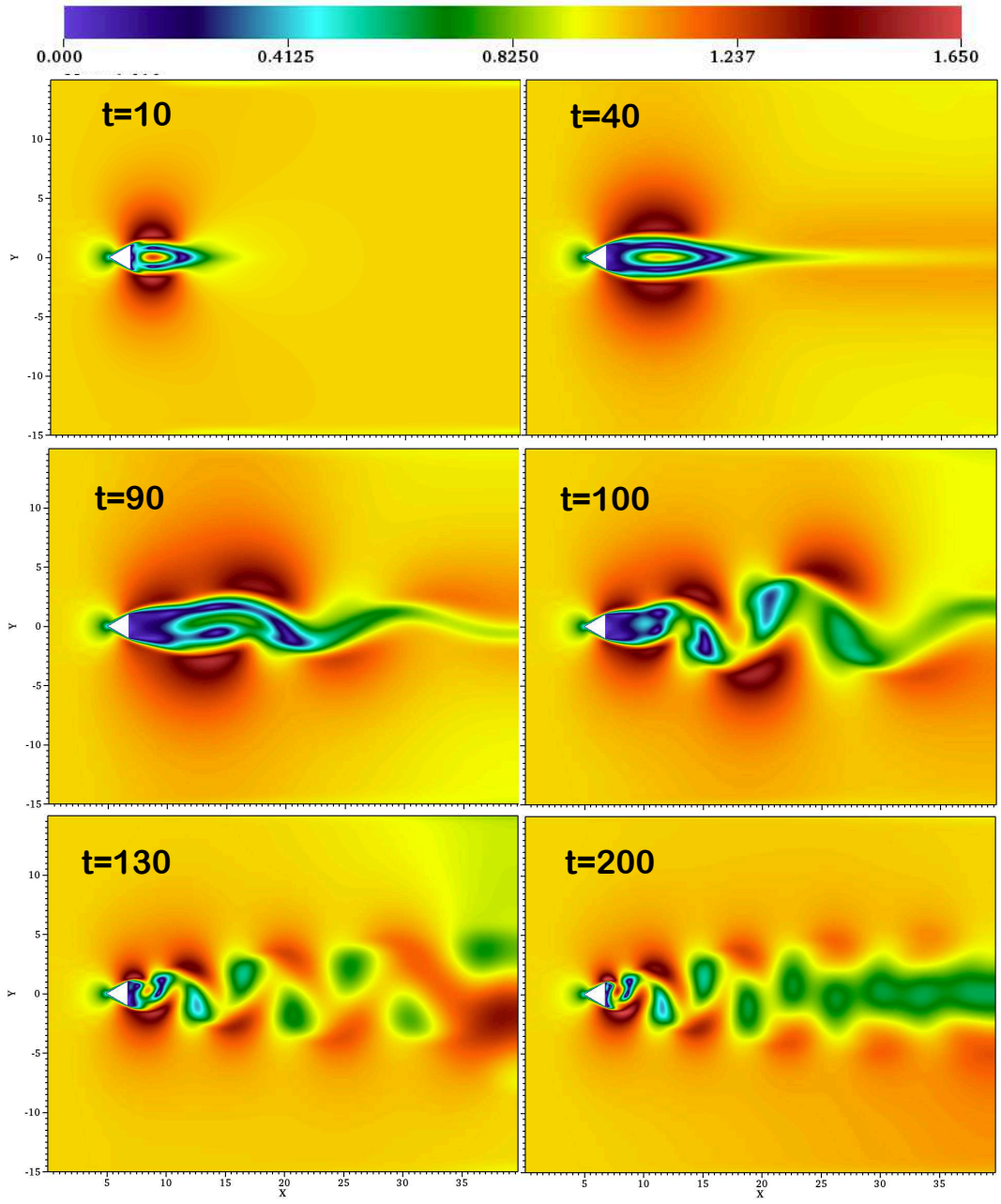


Fig. 5.46 : Vortex shedding behind the wedge, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh R2. Magnitude of the velocity vector.

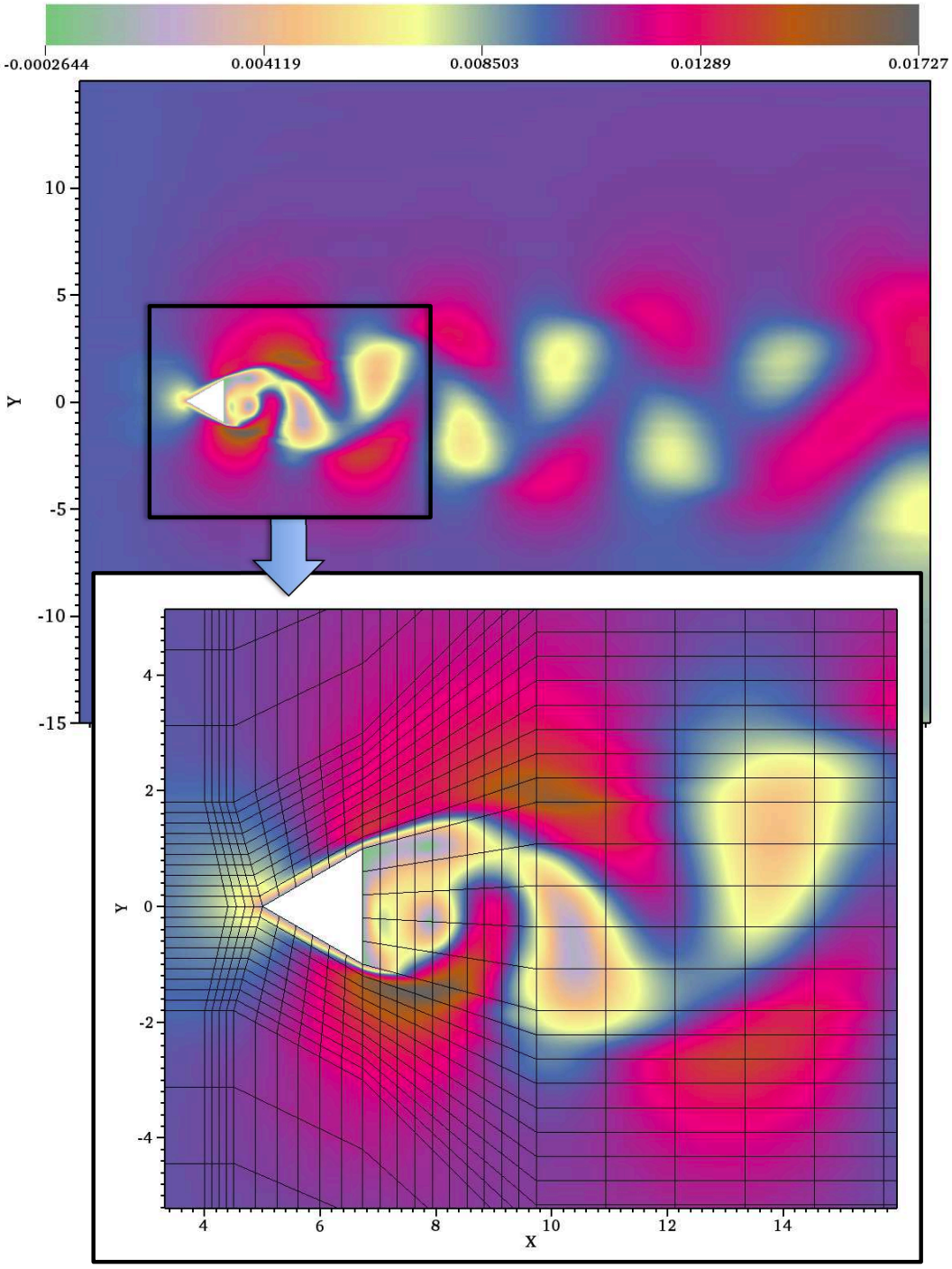


Fig. 5.47 : Vortex shedding behind the wedge, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh R1. Mach number field for $t = 130$.

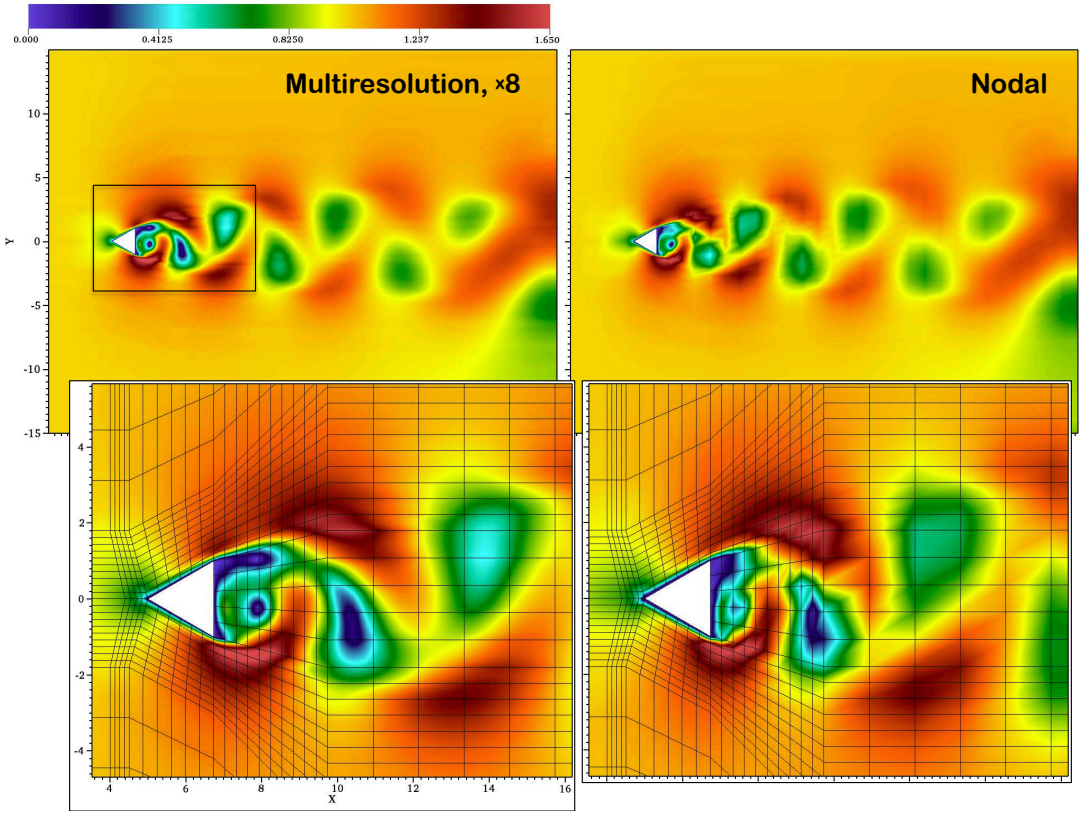


Fig. 5.48 : On visualization of high-order solution with multi-resolution option in VisIt. Vortex shedding behind the wedge, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh R1. Magnitude of the velocity vector for $t = 130$.

for definitions) ranging from 80 to 100, depending on scheme/resolution utilized.

First, we show results using the second-order $\mathrm{RDG}_{\mathrm{P_0P_1}}$. The flow was steady on the coarse meshes R1 and R2, see Figure 5.43. It seams that numerical diffusion in this scheme is strong enough to damp onset of any instability. We got unsteady vortex shedding when grid resolution was sufficiently high – as shown in Figure 5.44 for mesh R4. Note that no explicit triggering of instability is necessary.

The fourth-order solutions with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ are shown in Figures 5.45, and 5.46, for meshes R1 and R2, respectively. Comparing with the second-order solution in Figure 5.44 we notice that instability starts earlier (at $t \approx 75$ vs. $t \approx 100$ for $\mathrm{RDG}_{\mathrm{P_0P_1}}$). Also, with $\mathrm{RDG}_{\mathrm{P_0P_1}}$, the eddies are rather diffused, with only aproximately eight vortices (four pairs) present in the wake. The fourth-order solution produced approximately 10 eddies (five pairs) in the wake. This is related to better spatial resolution in the region immediately behind the wedge.

It is instructive to note that we are using recent "multi-resolution" option to visualize high-order solution representation in elements. In essence, VisIt allows to refine original elements, and high-order in-cell solution is mapped into the refined mesh. This allows to get much better images than those from using simple nodal solution mapping, as discussed in Section 5.6.1. The difference is particularly noticeable for low mesh resolution, as shown in Figure 5.48. In this particular visualization, we utilized MFEM output format, which is compatible with VisIt. Our six degrees of freedom are mapped into 25 gaussian points of Q4 elements, which are processed by VisIt to render multi-resolution images inside each element. In Figure 5.48, each element was subdivided on 8^2 sub-elements (original R1 mesh is also shown).

5.8 Thermally-Driven Natural Convection Flows

As our next test, we utilize a benchmark problem introduced by de Vahl Davis and Jones in [dJ83, de 83]. The flow in the unit square cavity is driven by gravity, due to the temperature difference $\Delta T = T_{\text{L}} - T_{\text{R}}$, applied at vertical walls. Horizontal walls are adiabatic. We use the 2-parameter EOS, Section C.6. Simulations are performed on the sequence of Rayleigh numbers $Ra = 10^3$, 10^4, 10^5 and 10^6, defined by eqs.(2.63). Prandtl number eq.(2.59) was fixed at 0.71. Mach number was varied from $\approx 10^{-4}$ (for $Ra = 10^3$) to 10^{-2} (for $Ra = 10^6$).

For buyoancy, we utilize the *Boussinesq* approximation, eq.(2.47).

Computations are done on the sequence of meshes 16×16, 32×32 and 64×64, with $\text{RDG}_{\text{P}_2\text{P}_3}$ and BDF_2 time discretization. Non-linear tolerances are set to 10^{-7}, for all solved equations. For all runs, steady-state solutions have been achieved, running with time steps corresponding to material CFL numbers in exceed of 50.

Computational results are presented in Figures 5.49-5.51. In Figure 5.49, we show temperature for different Ra numbers. The plotted isolines of temperature are in a very good agreement with the benchmark solutions given in [dJ83, de 83]. Steady-state velocity fields and streamline distributions are presented in Figure 5.50, for different Rayleigh numbers. Streamline fields are also consistent with those of [dJ83, de 83]. Finally, we plot solutions on the sequence of meshes, in Figure 5.51, which demonstrates space convergence of the solution. For this Ra number, mesh 32×32 is an adequate resolution with our fourth-order accurate scheme.

As an addition, we slightly modified the test configuration, performing similar simulations on non-square domain, described by Figure 5.2, using $\alpha = \beta = 20°$. This allows to test performance on irregular meshes. Computational results are shown in Figures 5.52 and 5.53, for three Ra numbers. As one can see, with the increase of Ra numbers, the boundary layers become thinner, and the overall vortex flow structure becomes more complex, forming secondary eddy, for $Ra = 10^6$.

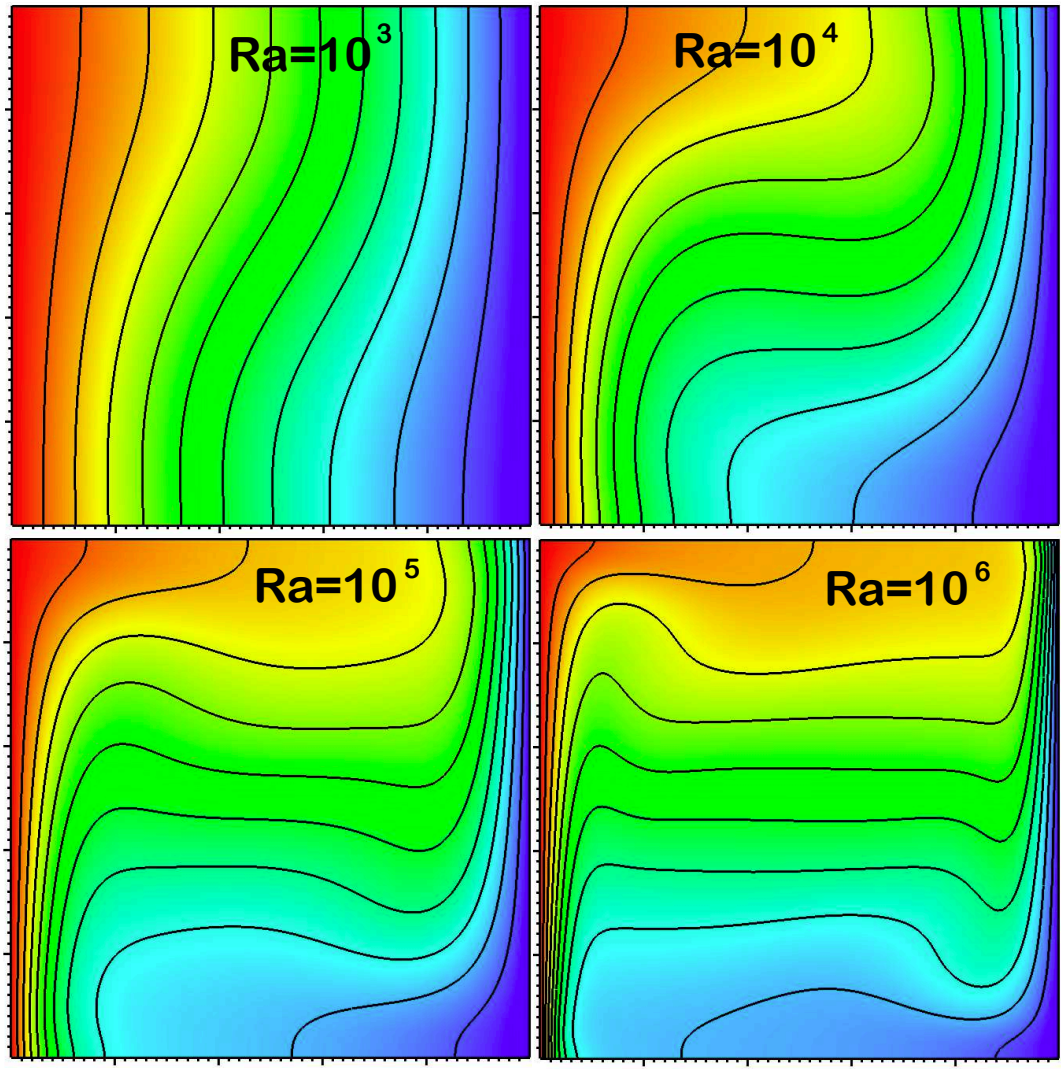


Fig. 5.49 : Temperature distributions for different Ra numbers, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh 32×32. 10 isolines.

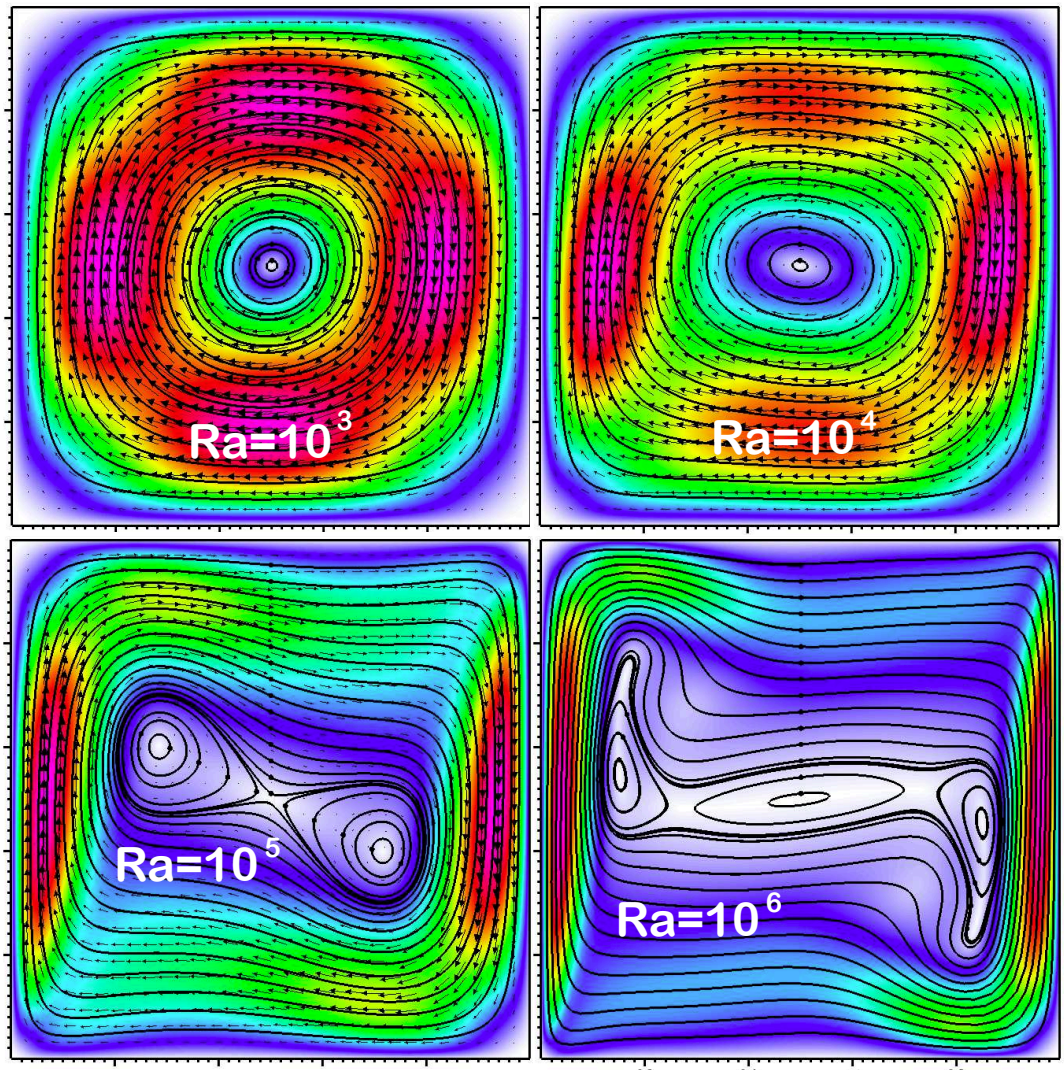


Fig. 5.50 : Velocity and streamline distributions for different Ra numbers, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh 32×32.

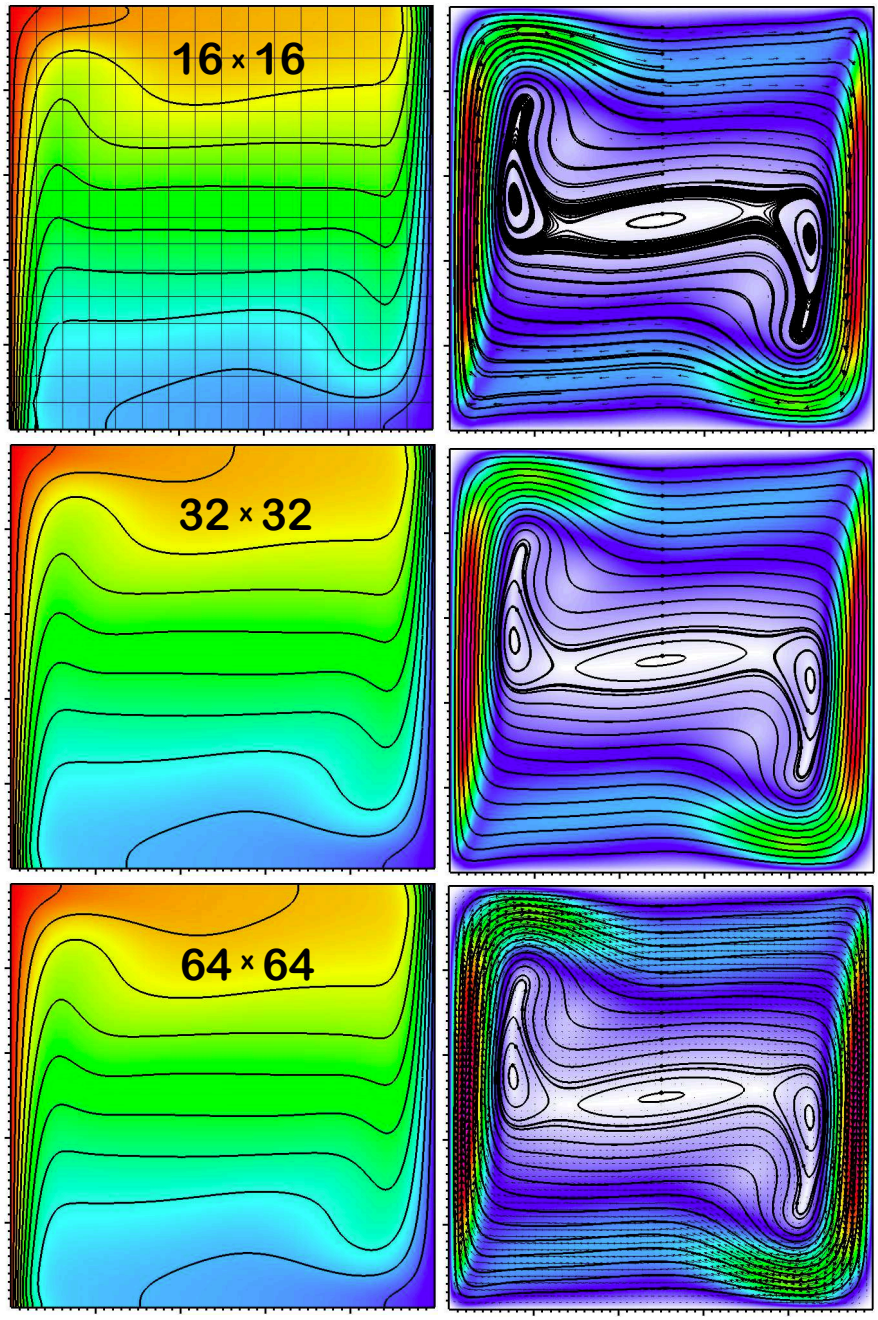


Fig. 5.51 : On convergence of temperature and velocity/streamline fields for $Ra = 10^6$, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$.

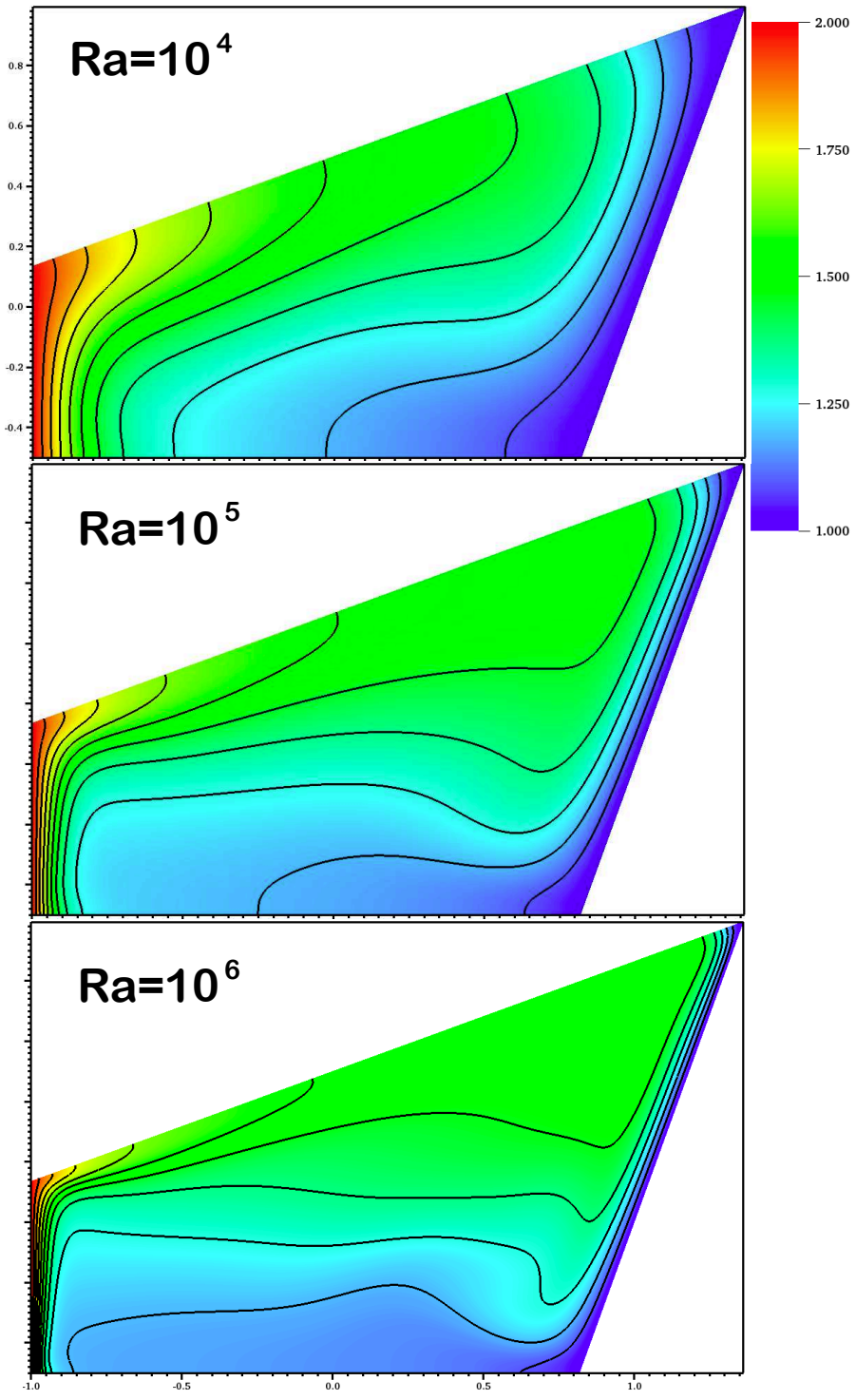


Fig. 5.52 : Temperature distributions for different Ra numbers, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh 64×32. 10 isolines.

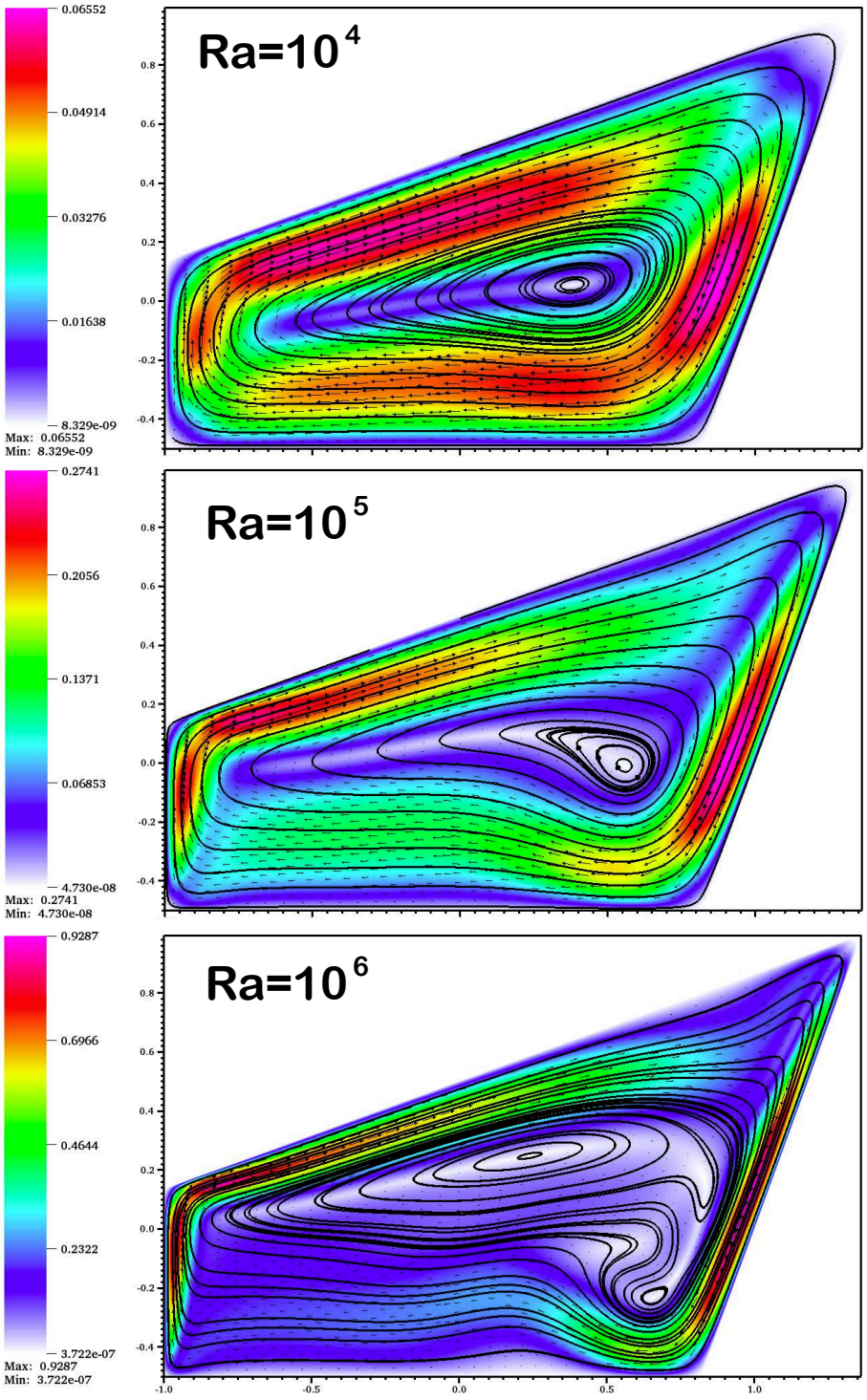


Fig. 5.53 : Velocity and streamline distributions for different Ra numbers, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh 64×32.

5.9 Rayleigh-Bénard Convection

In our next test, we investigate performance of our numerical algorithm for unstably-stratified natural convection flows. Computational domain is a 4×1 rectangular cavity, uniformly discretized on the sequence of meshes refined from 32×8 to 512×128. Vertical walls are adiabatic. The lower horizontal wall is heated (at const temperature), while the upper wall is cooled. Simulations are performed for $Pr = 0.1$ and the sequence of Ra numbers $4.5 \cdot 10^3$, $2.8 \cdot 10^4$ and $1.2 \cdot 10^6$ (see definitions in eqs.(**??**) and (**??**)). Similar simulations were performed in [Nou98], using commercial CFD code CFX4, based on the incompressible SIMPLE family of solvers. In current simulations, we kept Mach numbers $\approx 10^{-2}$. For low Ra numbers, $4.5\cdot10^3$ and $2.8\cdot10^4$, nearly steady-state solutions are obtained. For high $Ra = 1.2 \cdot 10^6$, no steady solution exists. All simulations are started with motionless constant-temperature field, slowly increasing temperature at the bottom to reach the needed temperature difference. We used BDF_2 time discretization, with time steps corresponding to material CFL ≈ 5, at quasi steady-state of the high-Ra-number test case. For low-Ra-number cases, steady-state solutions are reached with material CFL numbers in exceed of 100s. For all runs, nonlinear tolerance was set to 10^{-7}.

Computational results are presented in Figures 5.54-5.63. First, we show dependence of the solution fields on Ra number, in Figures 5.54 and 5.55. The solutions are steady for Ra numbers below 10^6. Above $Ra = 10^6$, the thermal cells/vortices become unstable, starting to "wiggle" at quasi steady-state, as shown in Figures 5.56 and 5.57.

Mesh convergence is shown in Figures 5.58 - 5.63. For low Ra number, the adequate solution can be obtained with just eight elements across the cavity, if the 4^{th}-order $\mathrm{RDG}_{\mathrm{P_2P_3}}$ scheme is utilized, Figures 5.58 and 5.59. Notice the slight sensitivity of the vortex position to the grid resolution, as the ascending plum is formed at the center of the cavity, for the mesh 32×8. This is different from the solution on the finer meshes, for which the vortices established at steady-state rotate in the opposite direction (i.e., at the center – we got a blob instead). This is typical of flows with instabilities, for which the final flow pattern is sensitive to modeling/discretization parameters.

Figures 5.60 and 5.61 depict grid convergence for high Ra number, using the fourth-order accurate $\mathrm{RDG}_{\mathrm{P_2P_3}}$ scheme. In this case, while we still observe de-

pendence of the vortex rotational direction on the mesh resolution, it is evident that can get a good solution with only eight elements in the vertical direction. All the flow features are captured, including the subtle "wiggling" of the plums, with a frequency consistent with high-fidelity runs. This is in contrast to the second-order $\mathrm{RDG}_{\mathrm{P_0P_1}}$ scheme, Figures 5.62 and 5.63, which exhibits stable steady-state patterns for meshes with a fewer than 64 elements in the vertical direction. With further mesh refinement, the finite volume $\mathrm{RDG}_{\mathrm{P_0P_1}}$ scheme resolves the proper flow pattern and plume "wiggling" effects.

It is instructive to note "broken" patterns for isolines of temperature, very visible for $\mathrm{RDG}_{\mathrm{P_0P_1}}$ on the coarse mesh. This is because of the picewise-linear representation of the solution on each element, which forms notable discontinuities at element faces, when mesh resolutions are poor. On the same mesh, piecewise-cubic $\mathrm{RDG}_{\mathrm{P_2P_3}}$ solution is much smoother.

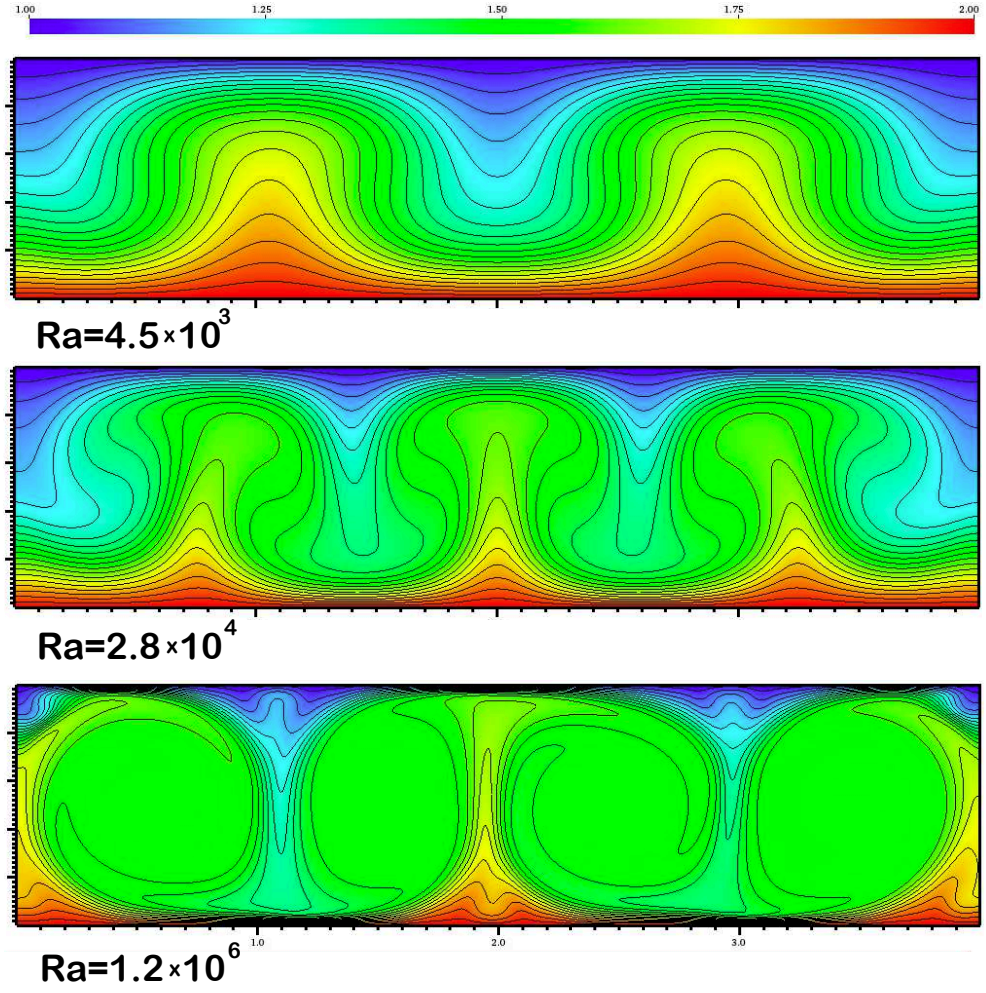


Fig. 5.54 : Temperature distributions for different Ra numbers, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh 128×32.

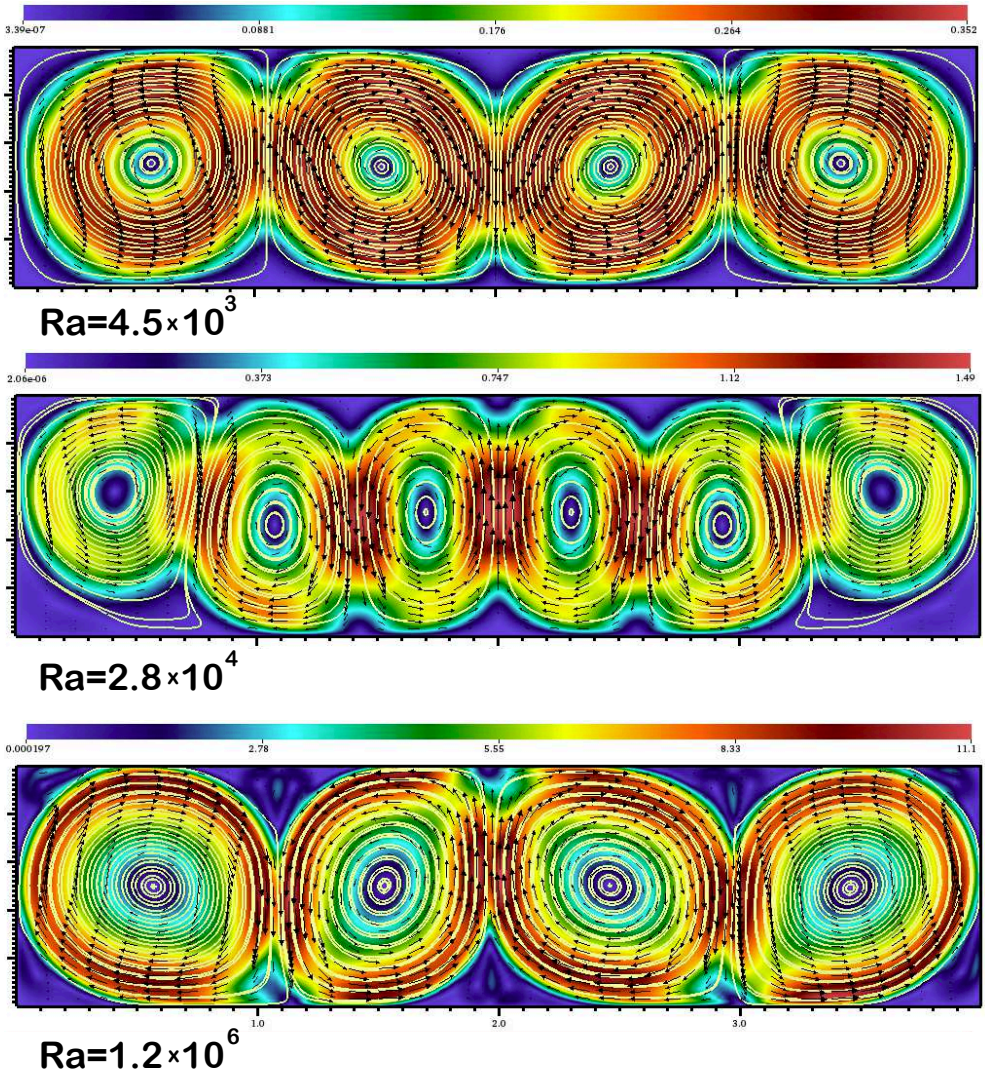


Fig. 5.55 : Velocity and streamline distributions for different Ra numbers, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh 128×32.

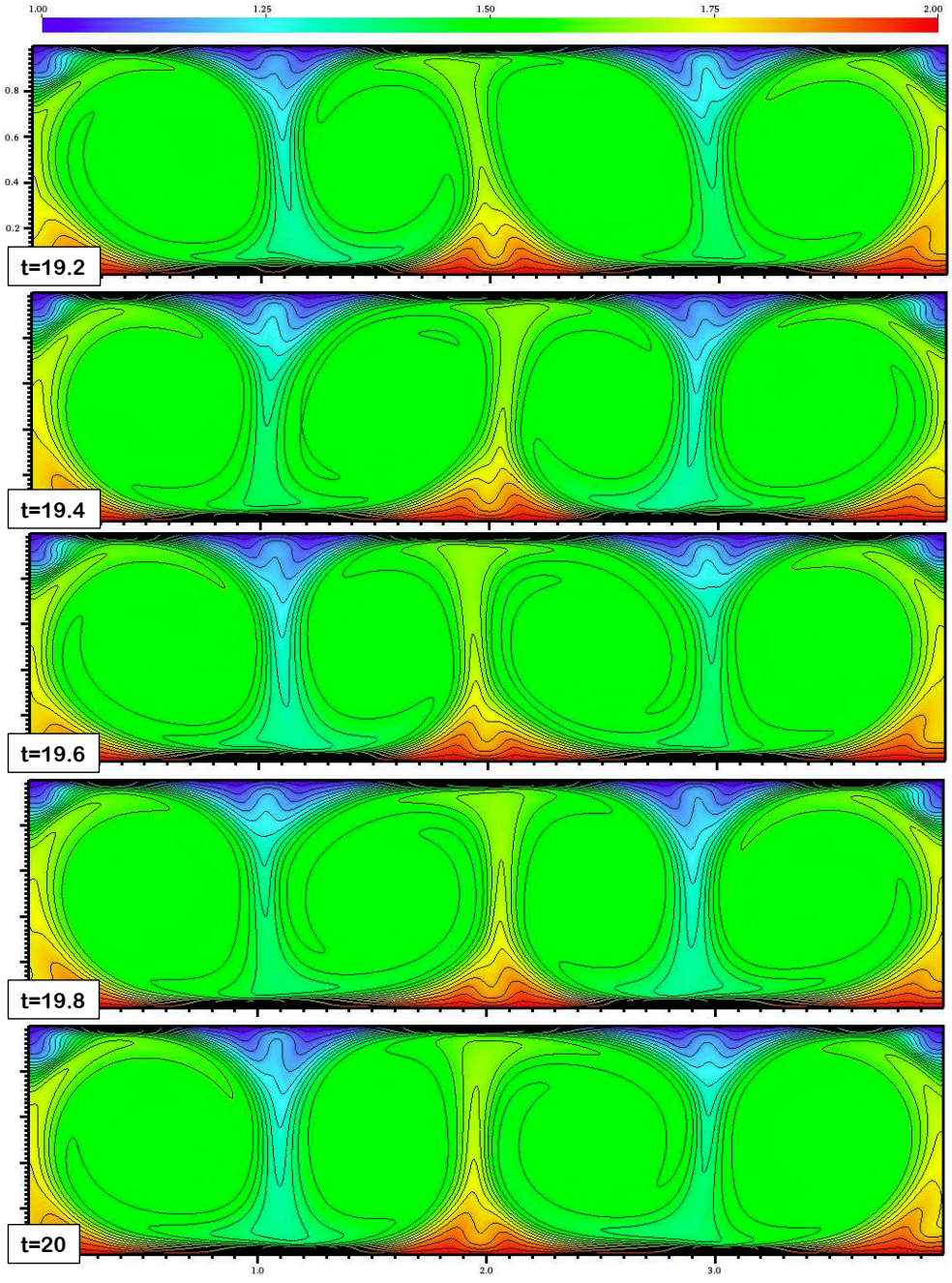


Fig. 5.56 : On oscillations of temperature field at quasi-steady-state, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh 128×32.

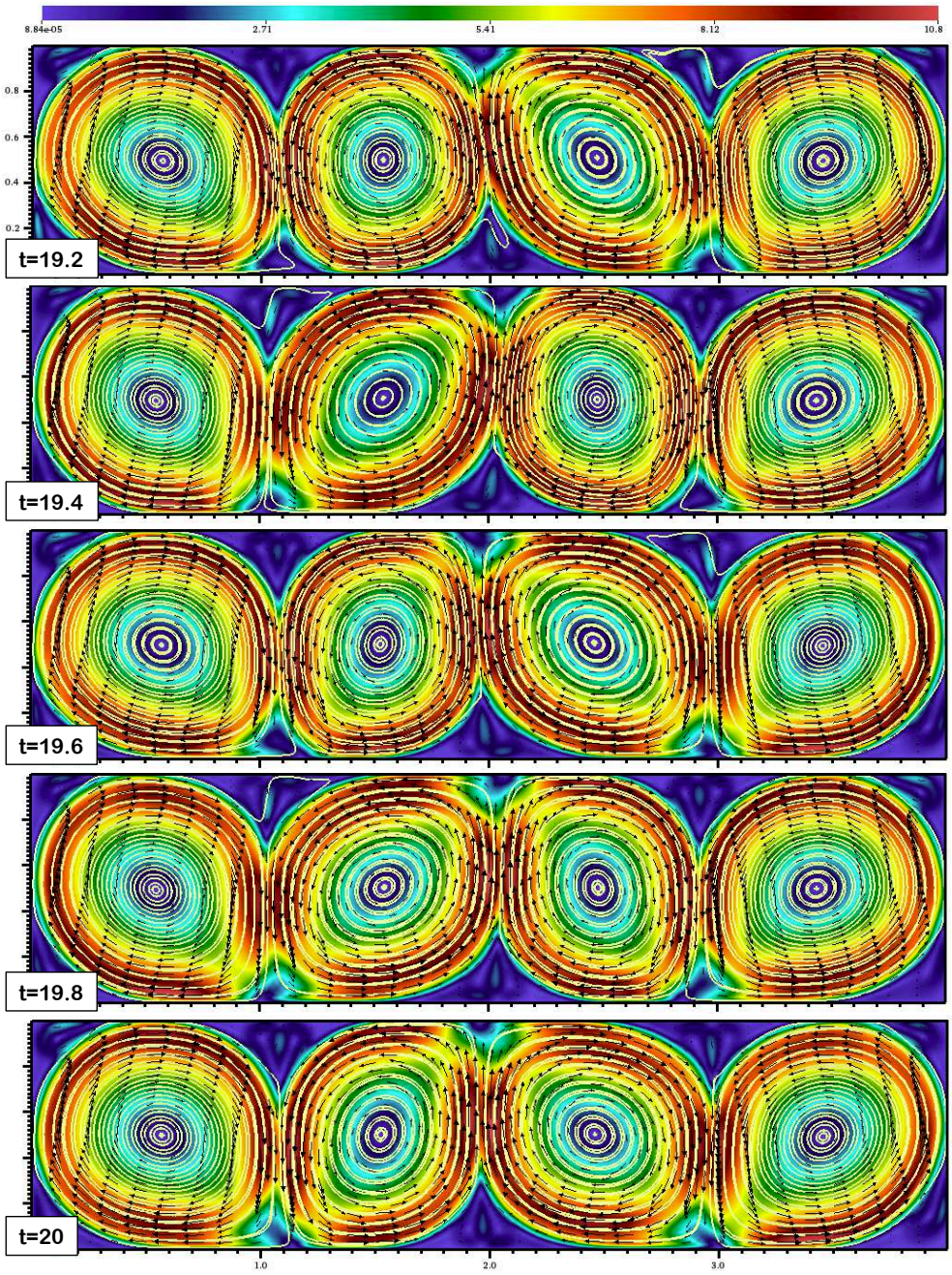


Fig. 5.57 : On oscillations of velocity and streamline fields at quasi-steady-state, with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on the mesh 128×32.

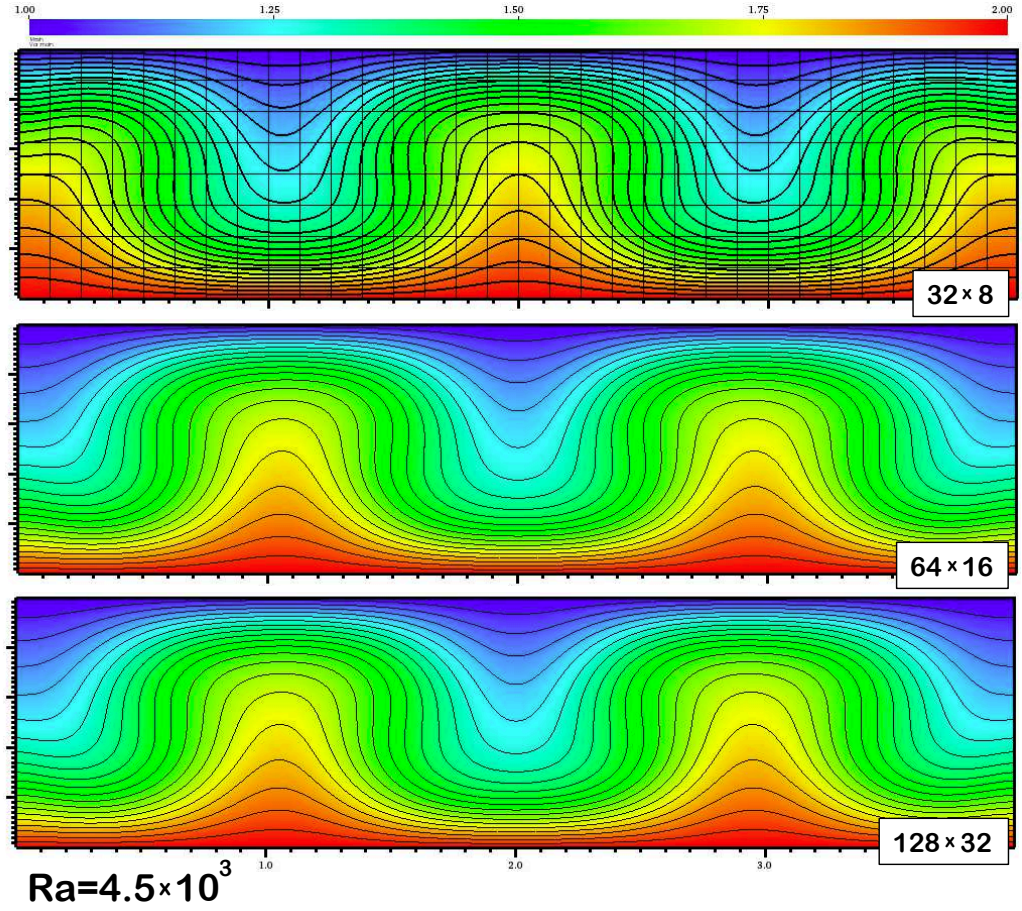


Fig. 5.58 : On mesh convergence of temperature field, using $\mathrm{RDG}_{\mathrm{P_2P_3}}$. $Ra = 4.5 \cdot 10^3$.

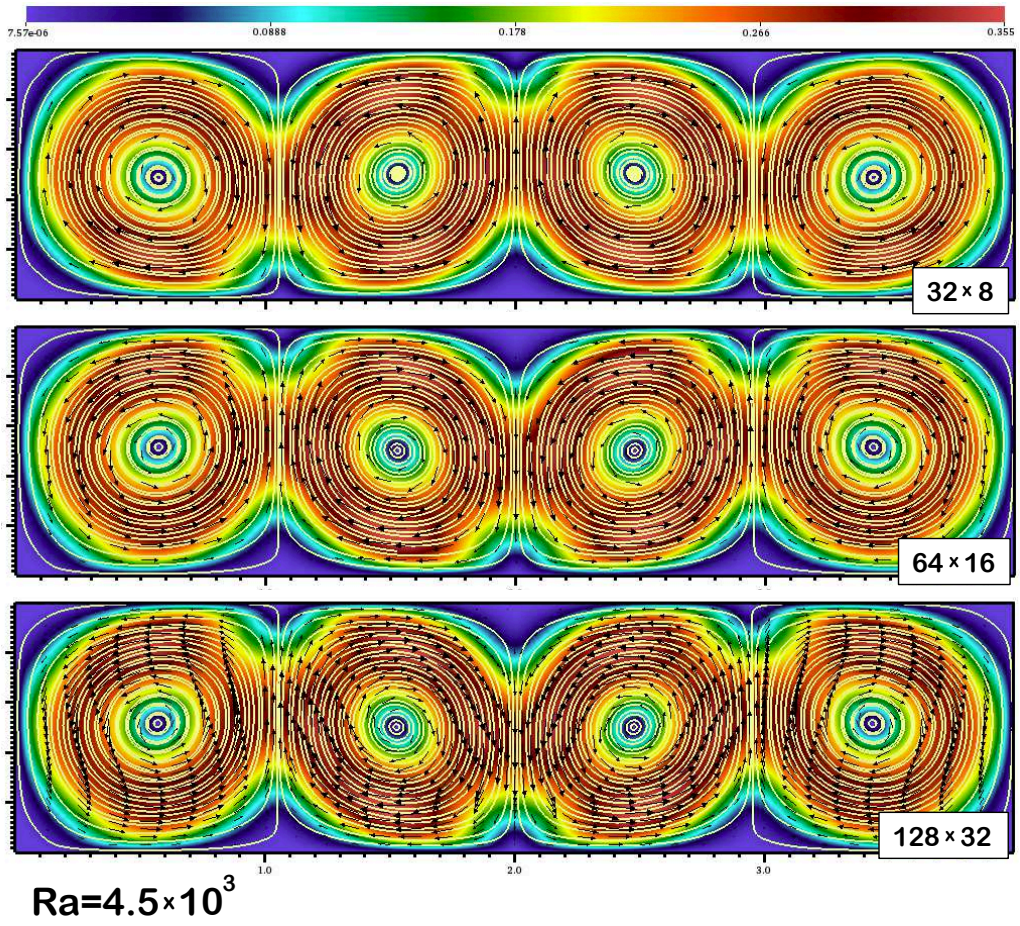


Fig. 5.59 : On mesh convergence of velocity/streamline fields, using $\mathrm{RDG}_{\mathrm{P_2P_3}}$. $Ra = 4.5 \cdot 10^3$.

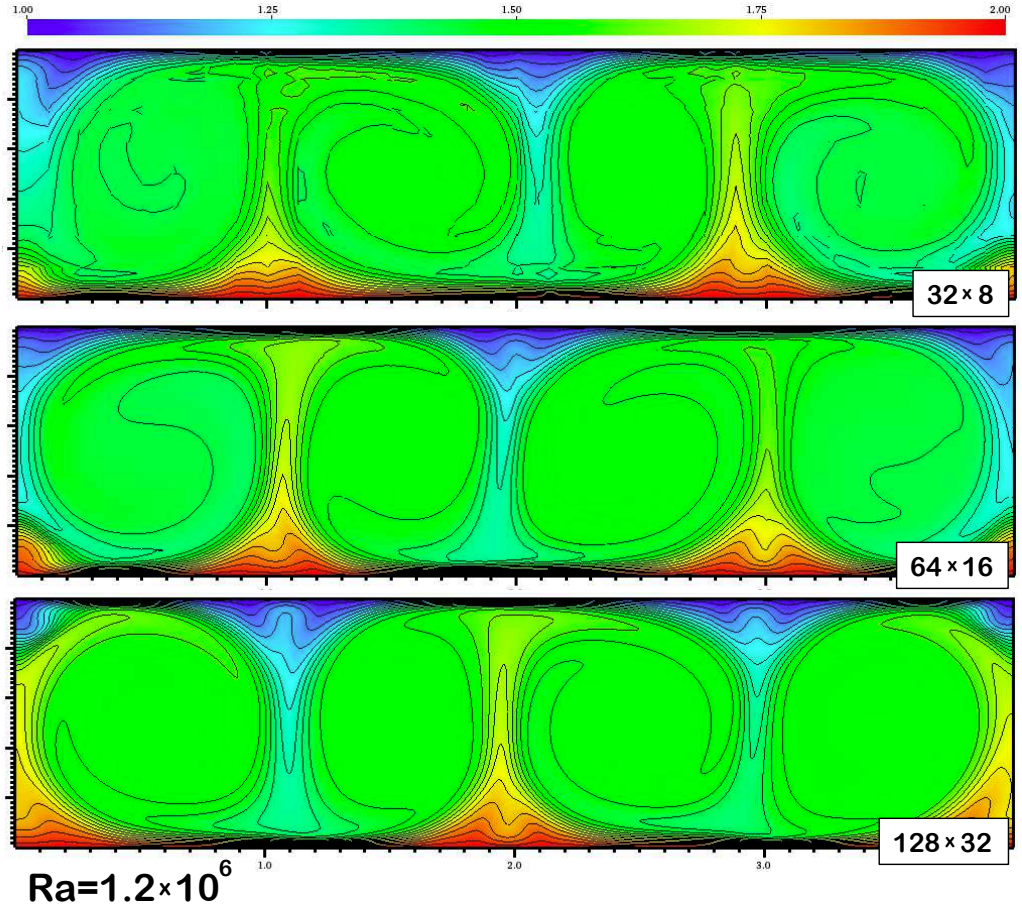


Fig. 5.60 : On mesh convergence of temperature field, using $\mathrm{RDG}_{\mathrm{P_2P_3}}$. $Ra = 1.2 \cdot 10^6$.

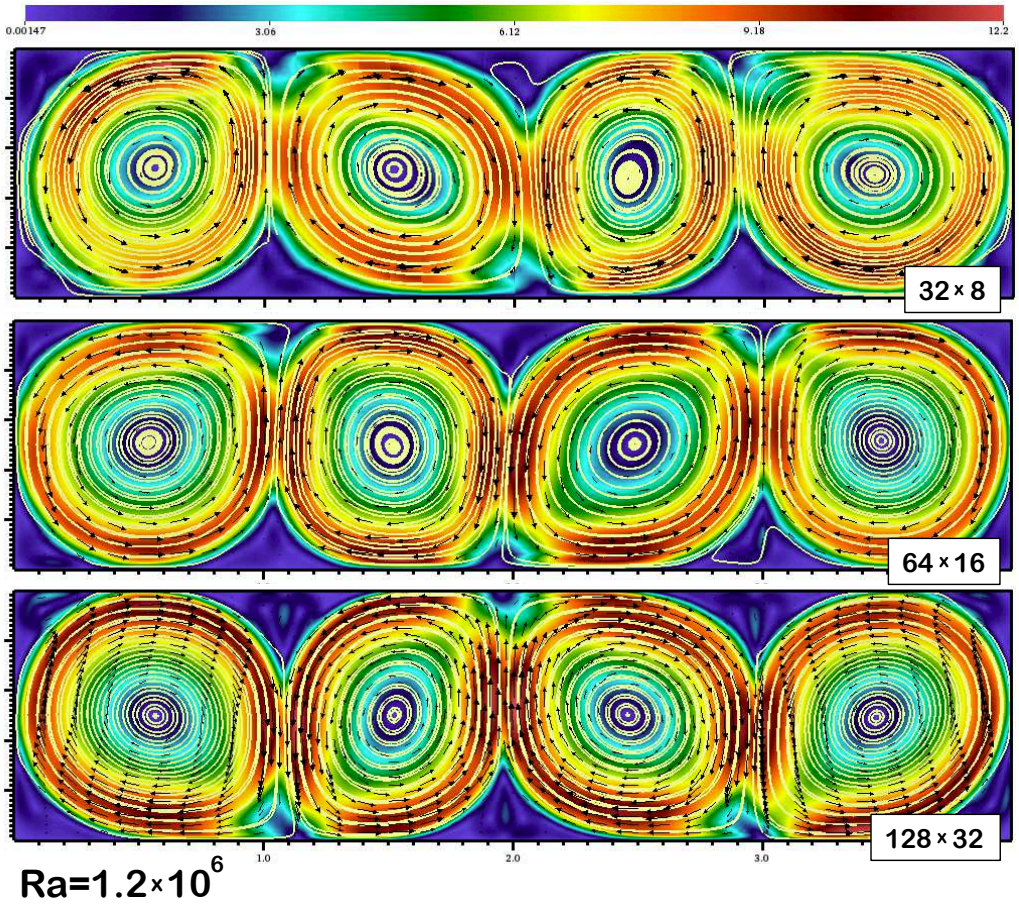


Fig. 5.61 : On mesh convergence of velocity/streamline fields, using $\mathrm{RDG}_{\mathrm{P_2P_3}}$. $Ra = 1.2 \cdot 10^6$.

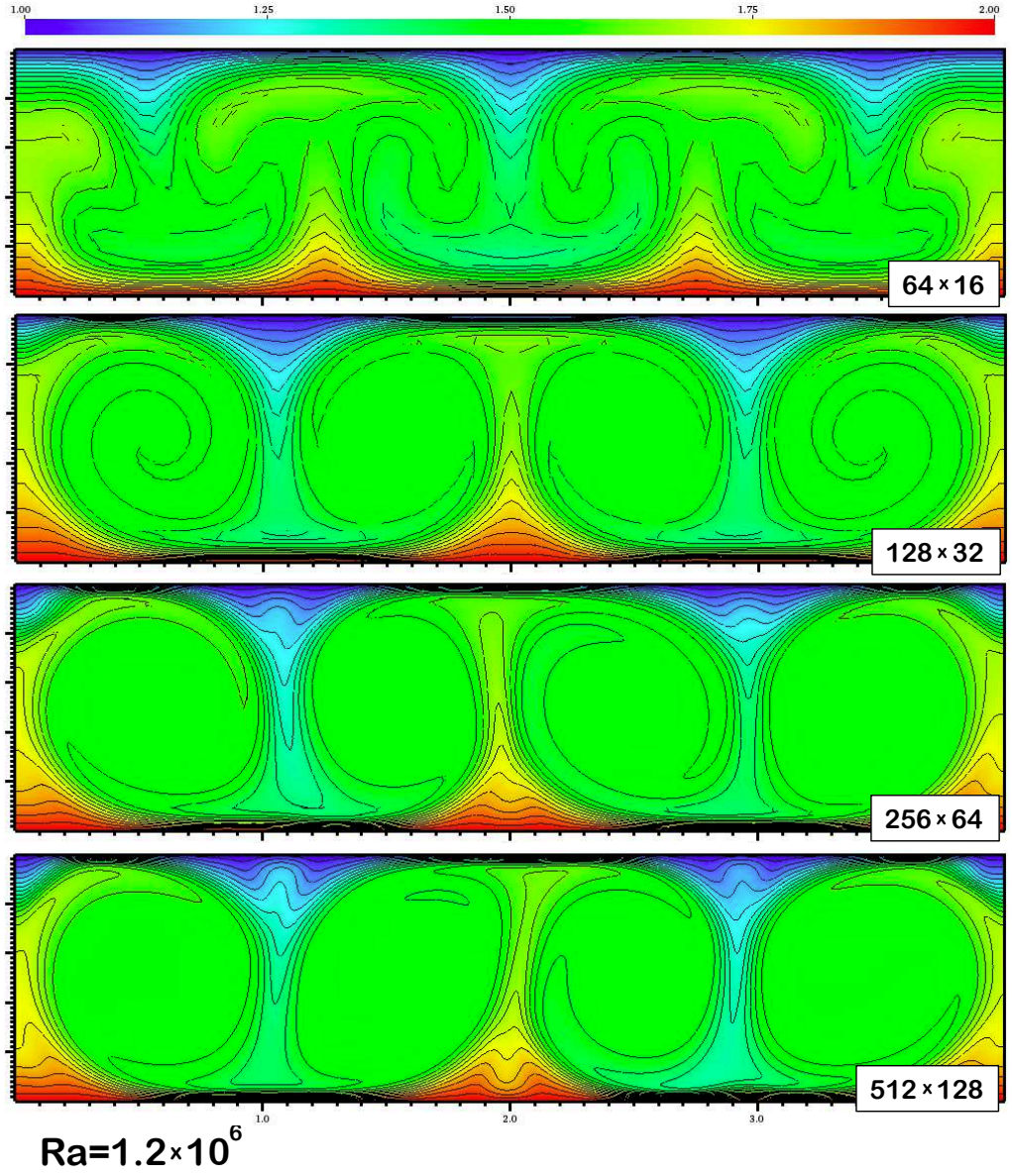


Fig. 5.62 : On mesh convergence of temperature field, using $\mathrm{RDG}_{\mathrm{P_0P_1}}$. $Ra = 1.2 \cdot 10^6$.

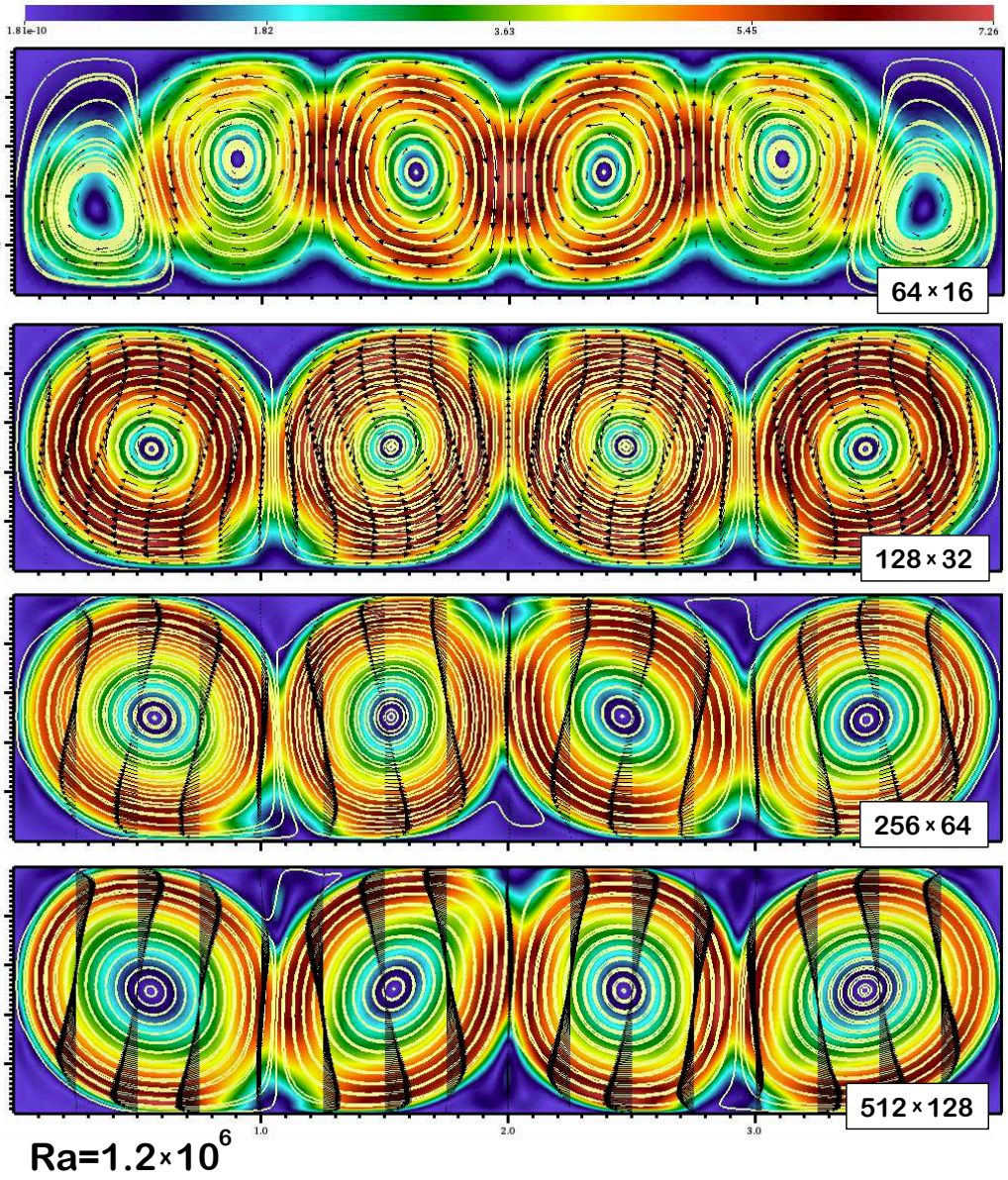


Fig. 5.63 : On mesh convergence of velocity/streamline fields, using $\mathrm{RDG}_{\mathrm{P_0P_1}}$. $Ra = 1.2 \cdot 10^6$.

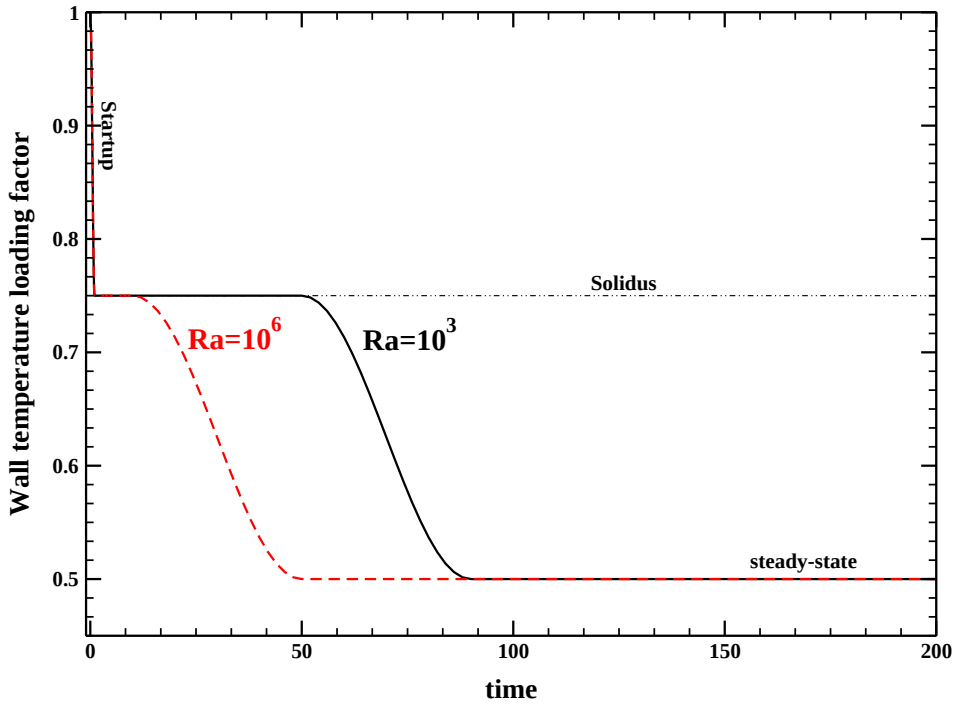


Fig. 5.64 : History of the left wall temperature loading factor.

5.10 Natural Convection with Phase Change

In our last numerical test, we investigate the method's performance for problems with solidification. The computational domain was an irregular rectangular box, as described by Figure 5.2, using $\alpha = \beta = 20°$. Two distinct test cases are investigated. The scaling parameters for these test cases are given in Table 5.5.

Initially, the pool was motionless, and the temperature was set well above the liquidus. In dimensionless units, $\hat{T}_{\text{init}} = 2$. The liquidus temperature was defined to be $\hat{T}_L = 1.5$, while the solidus temperature was $\hat{T}_S = 1.45$ for the low Ra-number case, and $\hat{T}_S = 1.4$ for the high Ra-number test problem. The transients are started with dropping the left wall temperature down to the liquidus temperature (smoothly, within one dimentionless time unit), keeping at this level for awhile – so that a nearly steady-state natural circulation is established, and then dropping the left wall temperature more, below the solidus, down to $\hat{T}_{\text{LFT}} = 1$, Figure 5.64. The right wall temperature was kept const, at $\hat{T}_{\text{RGT}} = 2$, well above the melting point. The top and bottom walls are adiabatic. This initiated a forma-

Table 5.5 : Scaling parameters for the tests

Parameter	**TEST-1**	**TEST-2**
Ra	10^3	10^6
Gr	10^4	10^7
Re	10^2	$3,162$
Pr	0.1	
Ste	4.854	

tion of the solid crust layer. We used the viscosity-based material strength model, as discussed in Section 2.3.3. The parameters of this model are $f_s = 10^2$, $\alpha = 10$ and $\omega = 2$, which corresponds to the variation of the viscosity factor as depicted in Figure 2.2.

Simulations are performed with the second-order BDF_2 time discretization, setting the non-linear tolerances at the level of 10^{-7} for the mass and momentum equations, and at 10^{-5} for the energy equation. A sample of the convergence for Newton iterations (typically converged within 4-5 iterations) is shown in Figure 5.65. The convergence curves for the mass and momentum are typical for quadratic rates (slightly curved upwards), at the first 3-4 iterations, before leveling off, at the asymptotic limit, when the round-off errors become important. The convergence rates for energy were slower, and leveling off earlier.

The dynamics of temperature, velocity, streamline fields, as well as melting front positions are shown in Figures 5.66-5.68. For both Ra numbers, the melting front advances slowly, until a steady-state position is achieved. In the case of low $Ra = 10^3$, a single vortex is established in the liquid pool. The boundary layer

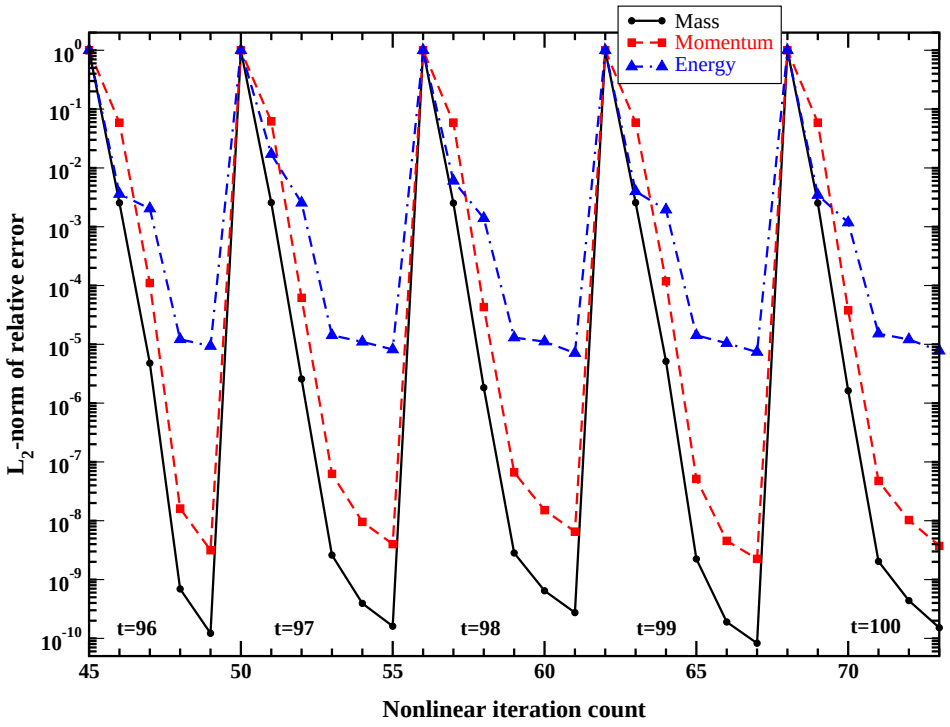


Fig. 5.65 : History of the non-linear convergence for the last five time steps of the simulation with $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh 128×64 and $Ra = 10^6$. $\mathrm{CFL}_{\mathrm{aco}} = 400$, $\mathrm{CFL}_{\mathrm{mat}} = 2$, $\mathrm{Fo}_{\mathrm{cond}} = 1.6$, $\mathrm{Fo}_{\mathrm{visc}} = 160$.

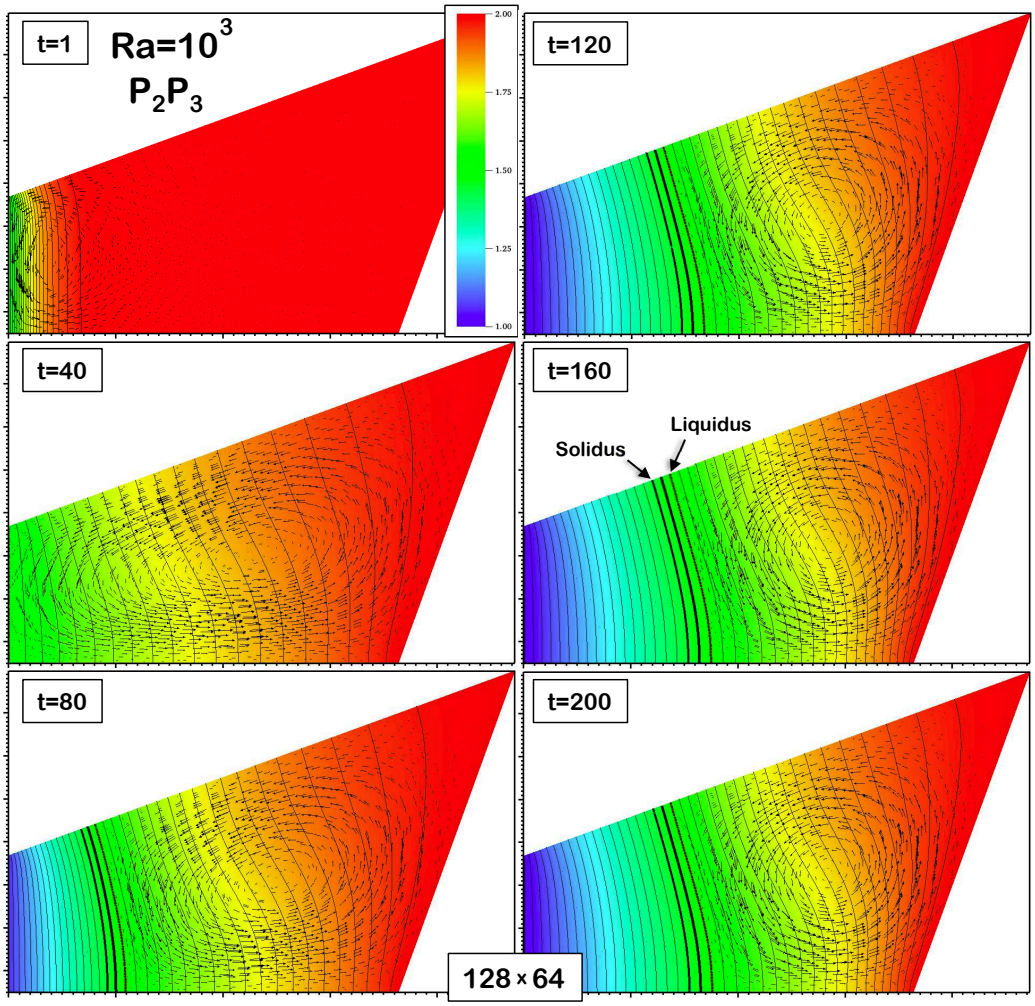


Fig. 5.66 : Dynamics of temperature field and melting front position, using $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh 128×64. $Ra = 10^3$.

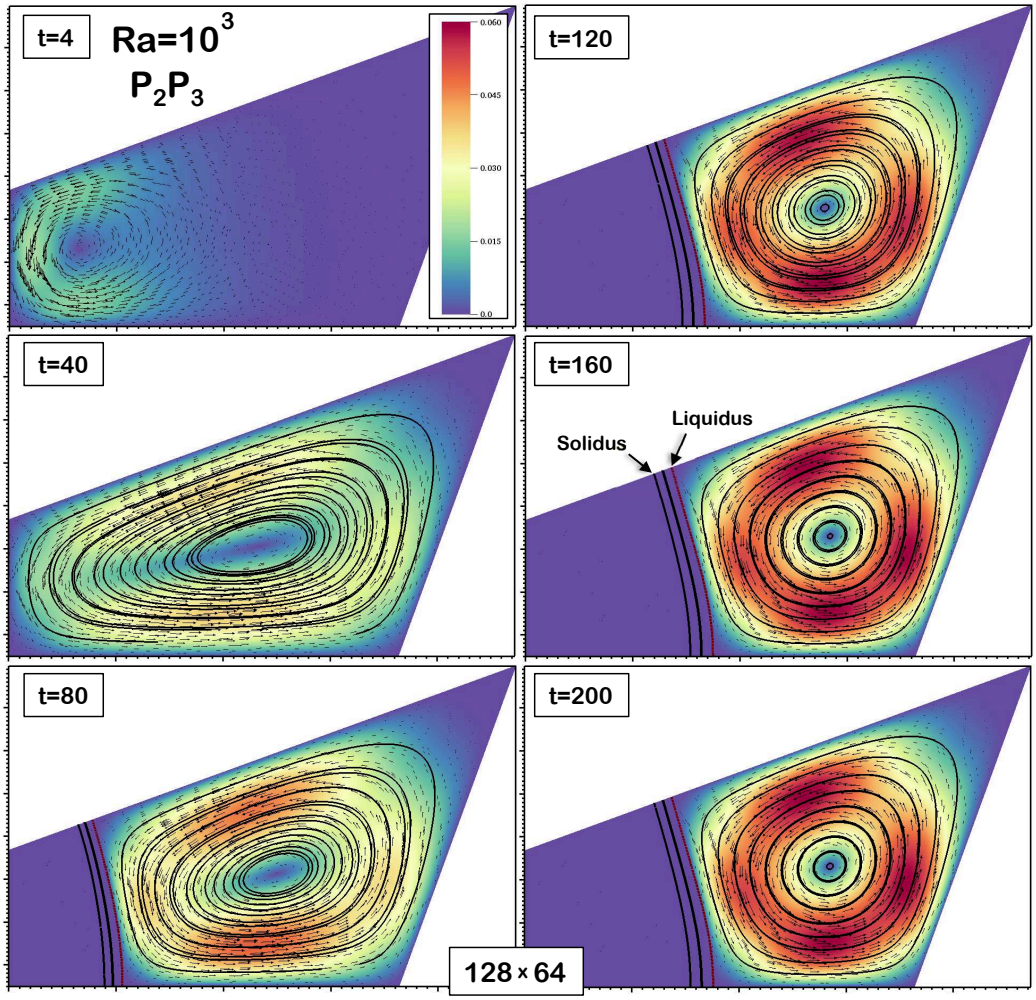


Fig. 5.67 : Dynamics of velocity/streamline fields and melting front position, using $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh 128×64. $Ra = 10^3$.

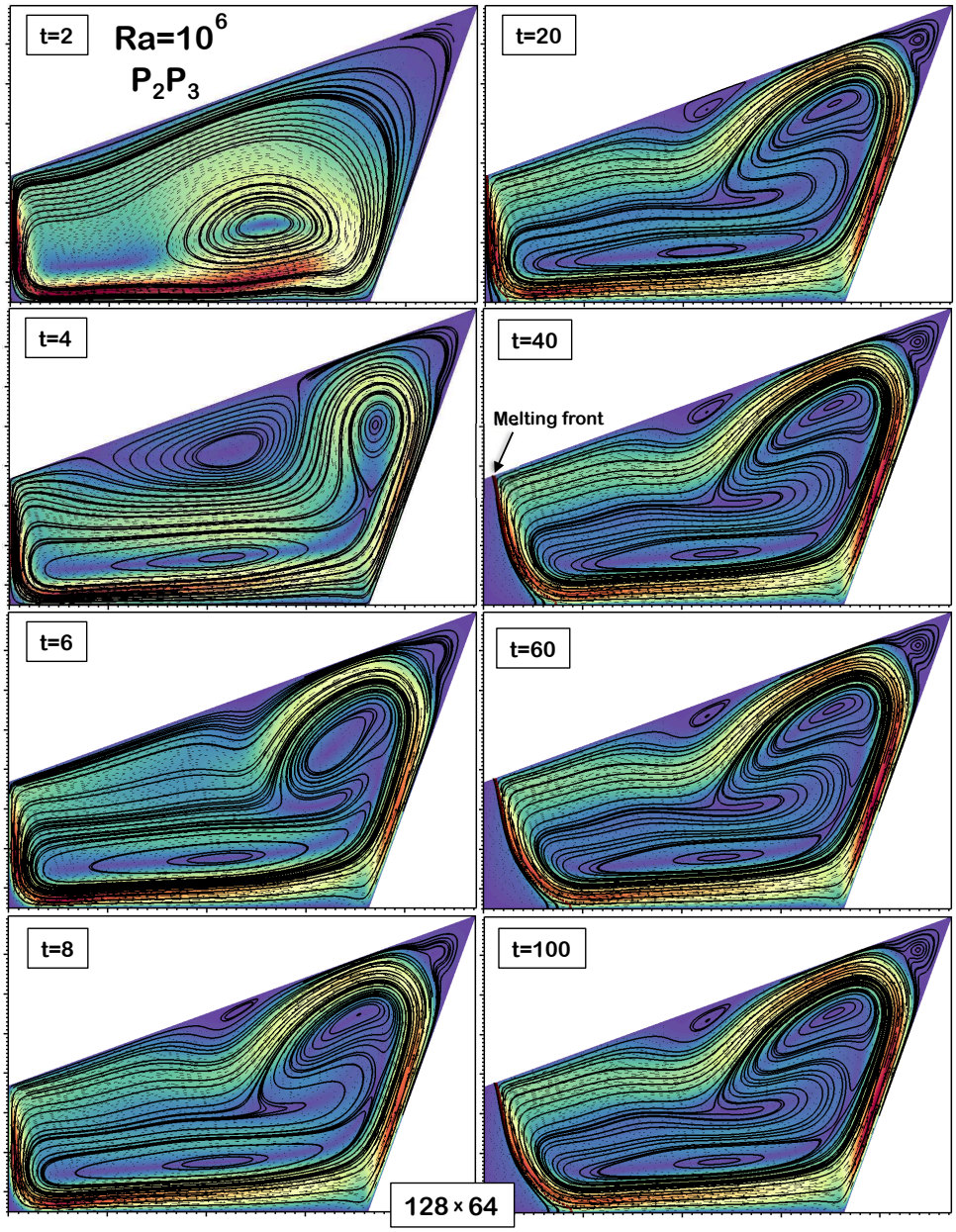


Fig. 5.68 : Dynamics of the velocity/streamline fields and melting front position, using $\mathrm{RDG}_{\mathrm{P_2P_3}}$ on mesh 128×64. $Ra = 10^6$.

near the melting front is relatively thick. Thus, it is rather easy to get an adequate solution with relatively course mesh and lower order spatial discretization, see Figures 5.69 - 5.72 for mesh convergence with both $\mathrm{RDG}_{\mathrm{P_0P_1}}$ and $\mathrm{RDG}_{\mathrm{P_2P_3}}$ schemes. This is a "never-mind" case, when the utilization of the high-order space discretization is not cost-effective.

It is very different for the high Ra number case. For $Ra = 10^6$, the established vortical flow pattern is more complex, forming five distinct eddies in the liquid pool. Also, the boundary layers and the "mushy" zone – are very thin. In Figures 5.73-5.75, we show the mesh convergence for the fourth-order accurate $\mathrm{RDG}_{\mathrm{P_2P_3}}$ scheme, on the sequence of meshes 32×16, 64×32 and 128×64. It is evident, that even on the coarsest mesh, our method represents flow pattern adequately, with all five vortices resolved (the upper-right corner vortex is actually broken into two, due to extremely coarse in this corner). For the $\mathrm{RDG}_{\mathrm{P_2P_3}}$, we would consider the mesh 64×32 to be "good-enough", based on these "viewgraph-norm" comparisons.

The mesh convergence for $\mathrm{RDG}_{\mathrm{P_0P_1}}$ is demonstrated in Figures 5.76-5.78. It can be seen that the second-order discretization errors tend to suppress resolution of smaller vortical structures. Only with the mesh as high as 512×256 we are getting a full resolution of five vortical structures. Comparing to $\mathrm{RDG}_{\mathrm{P_2P_3}}$, this requires ten times more degrees of freedom, which effectively corresponds to significantly larger linear algebra matrices to invert, and significantly larger CFL numbers involved (thus, stiffer underlying linear solver). In terms of CPU time, we do not get ten-fold speed-up, because the $\mathrm{RDG}_{\mathrm{P_2P_3}}$ is more expensive per DoF, than the $\mathrm{RDG}_{\mathrm{P_2P_3}}$ – but still a better (2-3 times) performance. It is instructive to note here, that higher-order schemes allow to better utilize future trends in hardware architecture, including GPU computing and threading, which would make advantage of the $\mathrm{RDG}_{\mathrm{P_2P_3}}$ scheme even more evident.

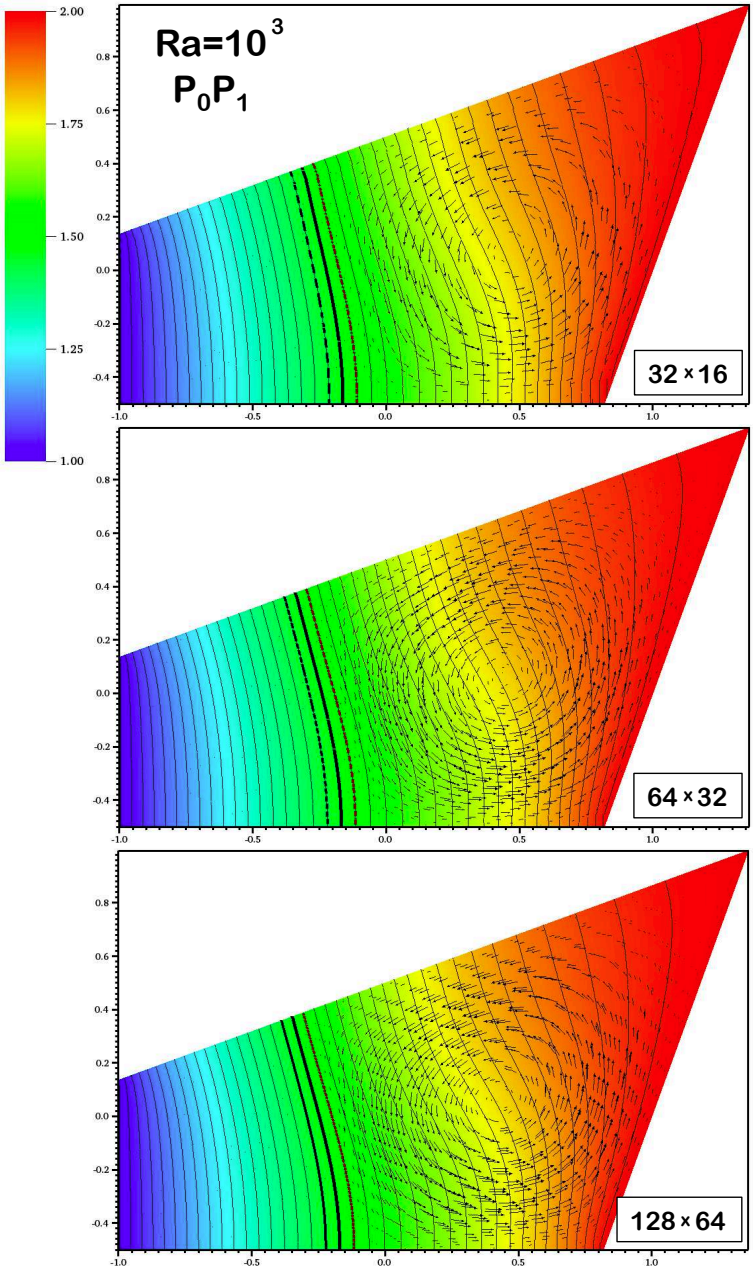


Fig. 5.69 : Mesh convergence of temperature field for $\mathrm{RDG}_{\mathrm{P_0P_1}}$. $Ra = 10^3$.

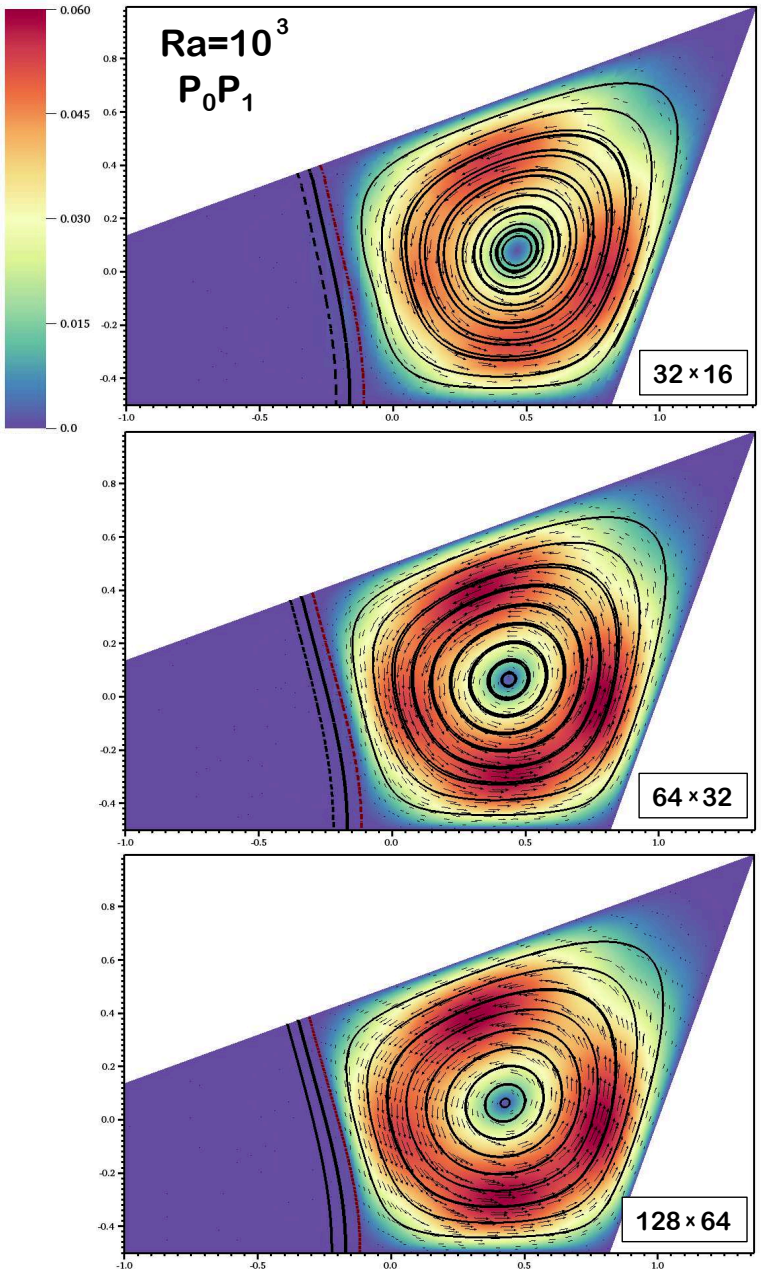


Fig. 5.70 : Mesh convergence of velocity/streamline fields for $\mathrm{RDG}_{\mathrm{P_0P_1}}$. $Ra = 10^3$.

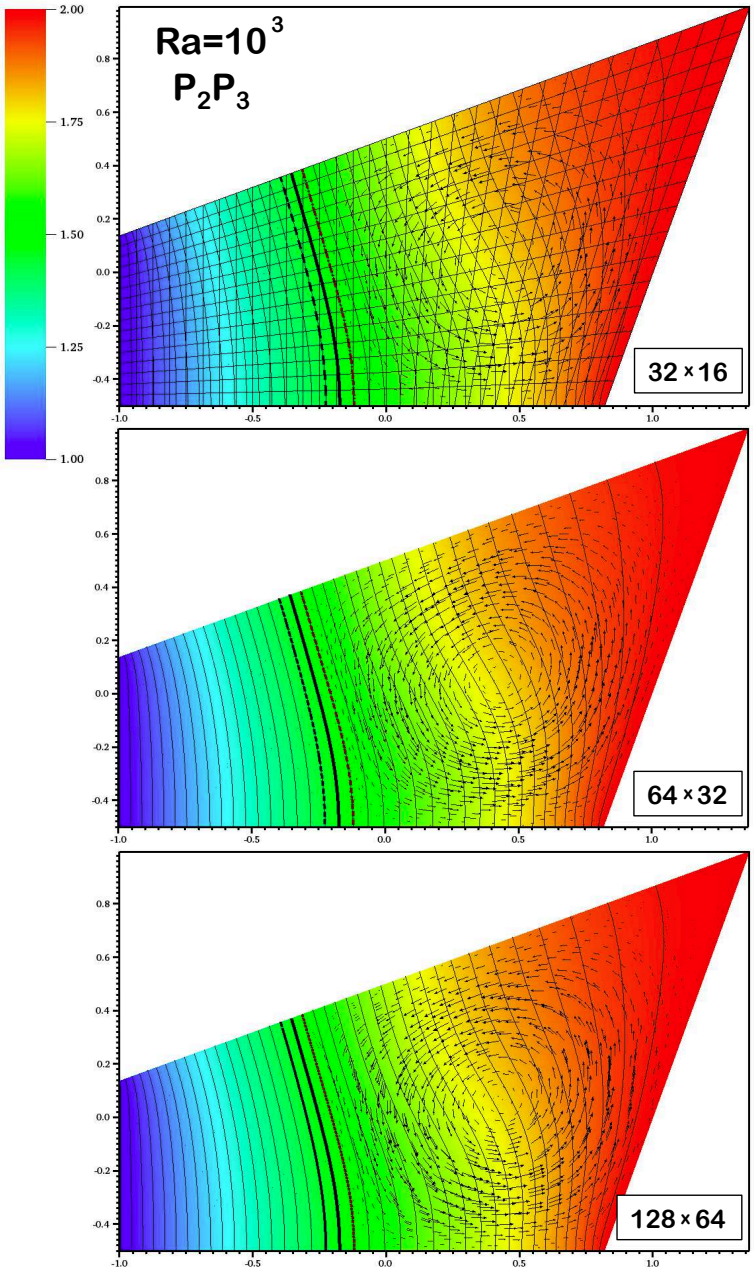


Fig. 5.71 : Mesh convergence of temperature field for $\mathrm{RDG}_{\mathrm{P_2P_3}}$. $Ra = 10^3$.

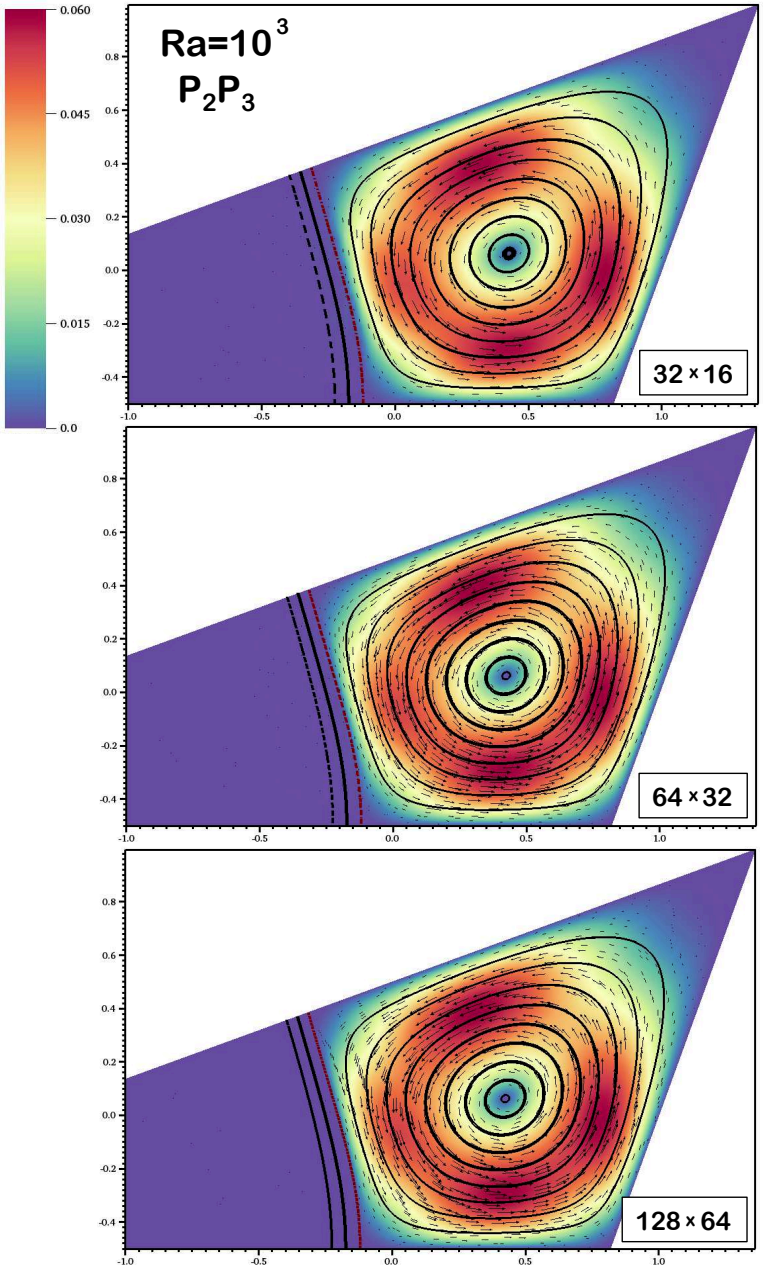


Fig. 5.72 : Mesh convergence of velocity/streamline fields for $\mathrm{RDG}_{\mathrm{P_2P_3}}$. $Ra = 10^3$.

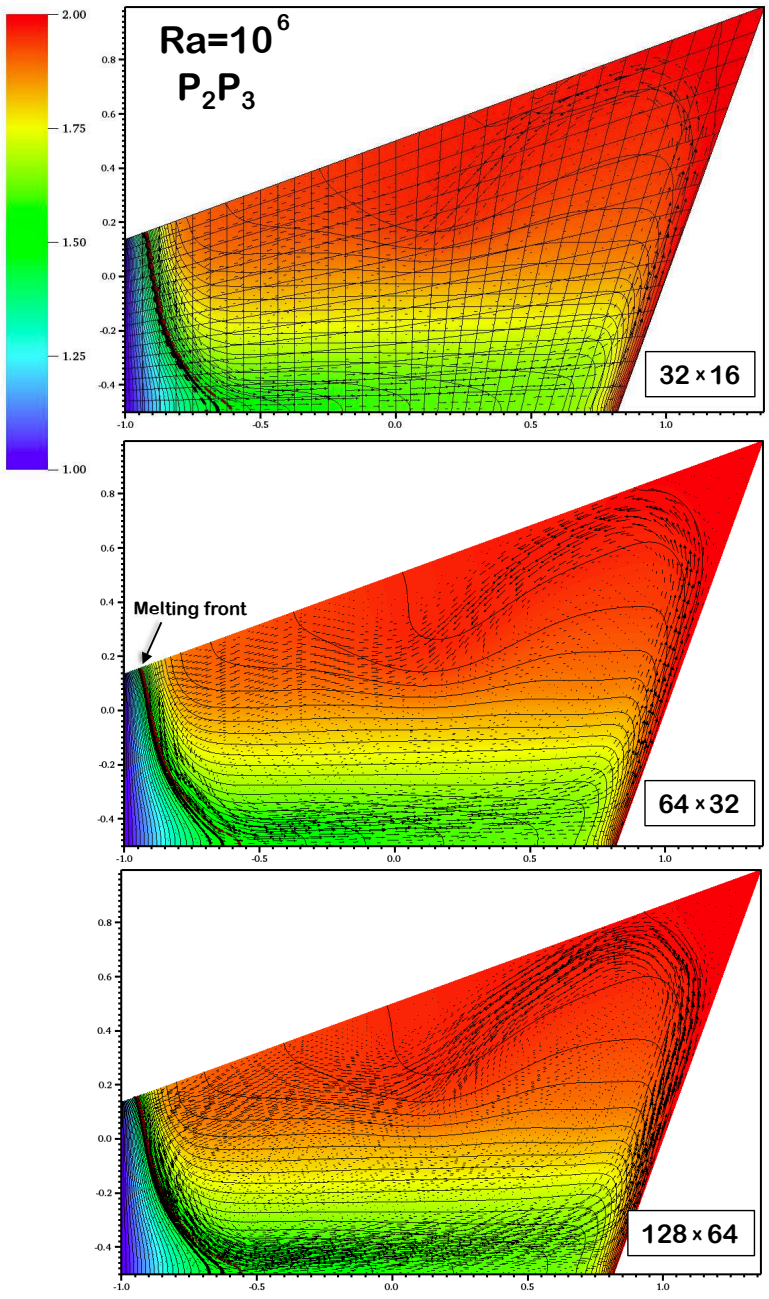


Fig. 5.73 : Mesh convergence of temperature field for $\mathrm{RDG}_{\mathrm{P_2P_3}}$. $Ra = 10^6$.

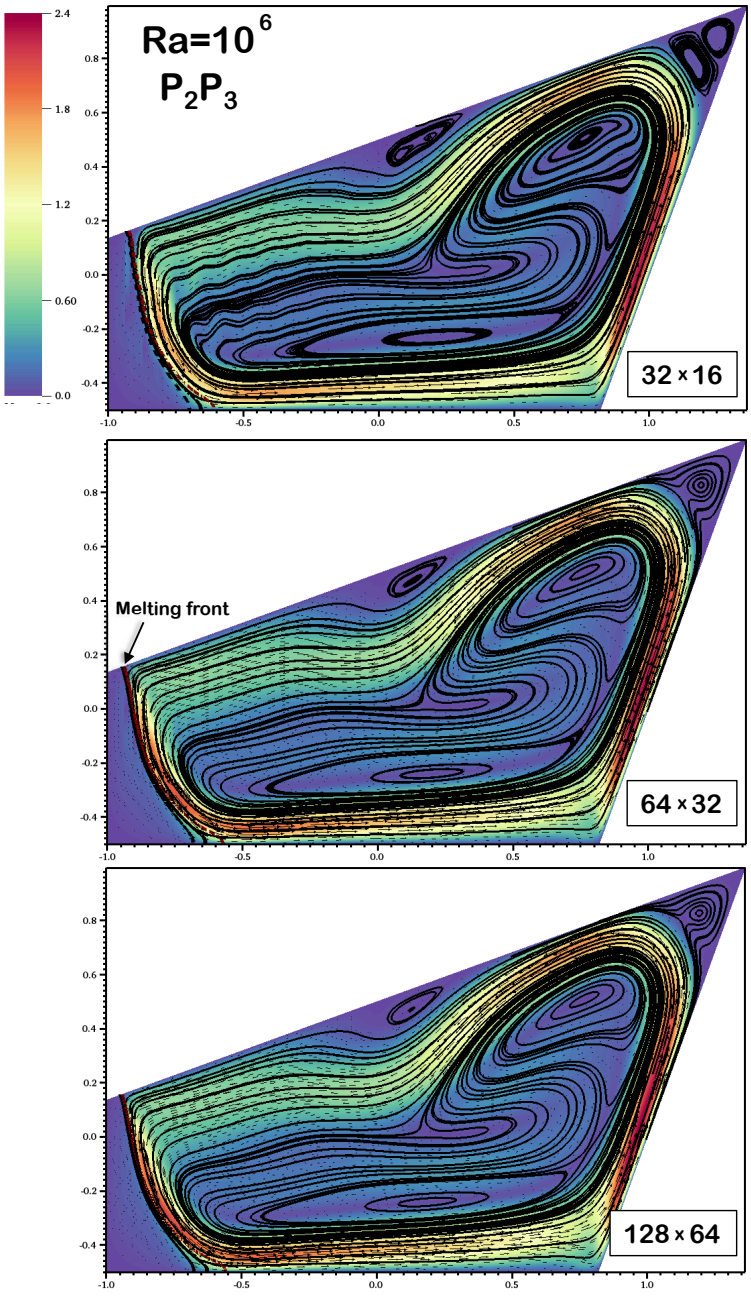


Fig. 5.74 : Mesh convergence of velocity/streamline fields for $\mathrm{RDG}_{\mathrm{P_2P_3}}$. $Ra = 10^6$.

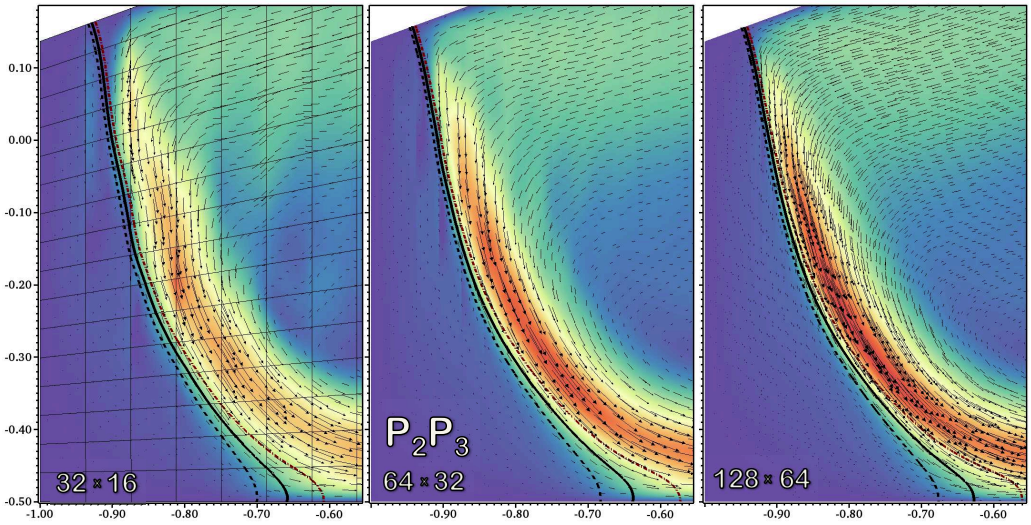


Fig. 5.75 : Mesh convergence of velocity field and melting front position, for $\mathrm{RDG}_{\mathrm{P_2P_3}}$. $Ra = 10^6$.

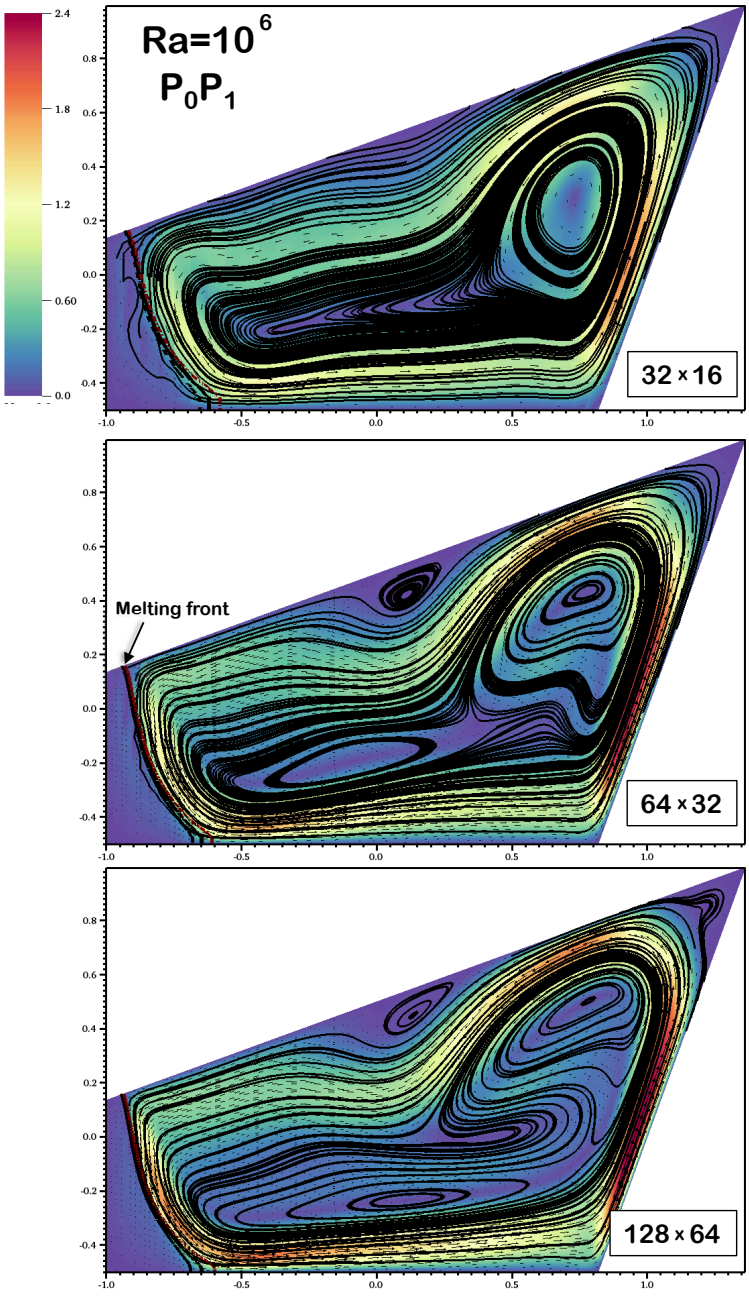


Fig. 5.76 : Mesh convergence of velocity/streamline fields for $\mathrm{RDG}_{\mathrm{P_0P_1}}$. $Ra = 10^6$.

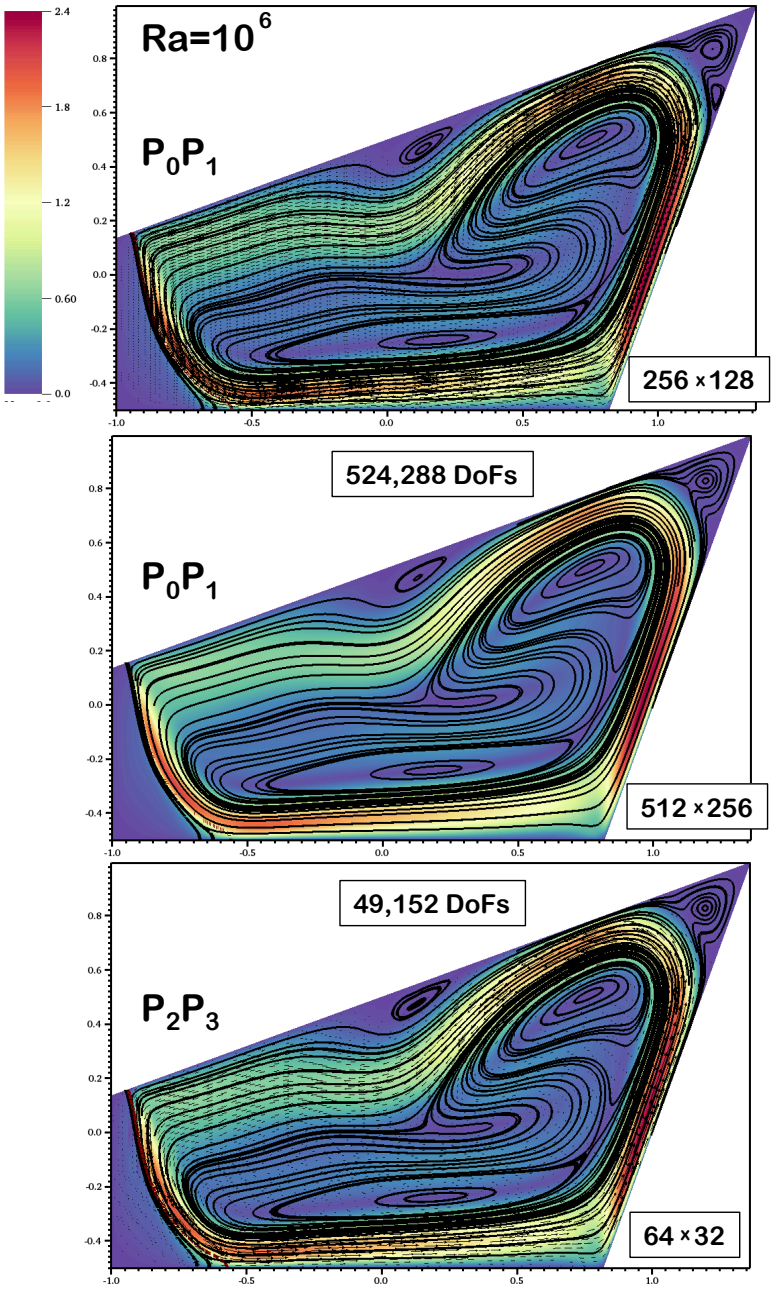


Fig. 5.77 : Comparison of velocity/streamline fields for "converged" mesh. $\mathrm{RDG}_{\mathrm{P_0P_1}}$ vs. $\mathrm{RDG}_{\mathrm{P_2P_3}}$. $Ra = 10^6$.

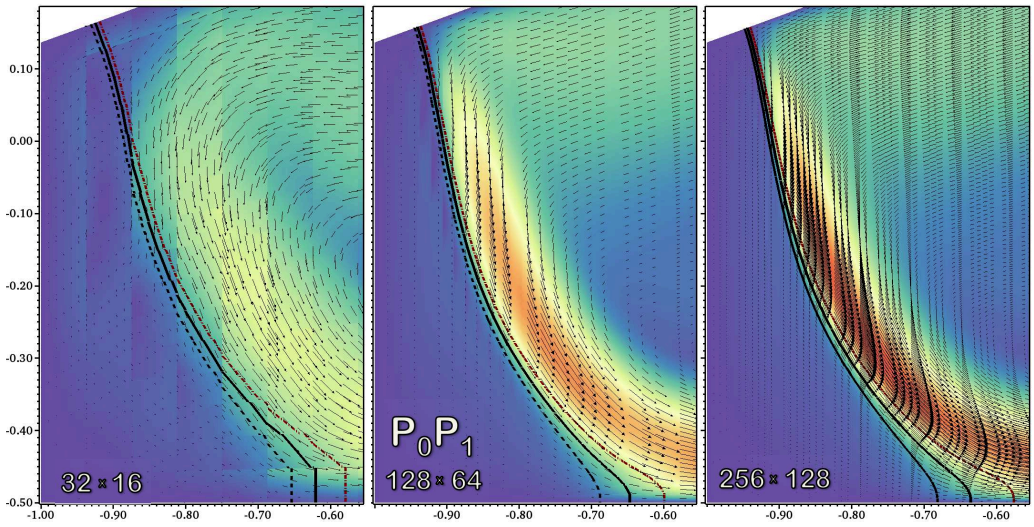


Fig. 5.78 : Mesh convergence of velocity field and melting front position, for $\mathrm{RDG}_{\mathrm{P_0P_1}}$. $Ra = 10^6$.

This Page is Intentionally Left Blank

Chapter 6

Concluding Remarks

In this study, we introduced Discontinuous Galerkin method based on the Least-Squares reconstruction and orthogonal basis functions. The method is generally a subclass of the Method of Mean Weighted Residuals (MWR), in which we solve for degrees of freedom in a piecewise-polynomial solution representation, minimizing residuals satisfying underlying governing equations. There are two main technical contributions of the work - *a)* Tensor-product Legendre polynomials as basis functions, combined with inverse Jacobian weighted test functions, ensuring orthogonality of the mass matrix; and *b)* An ability of solving for primitive variables, chosen on the base of better conditioning/solvability of the underlying governing equations. These features enable robust combination of the RDG with Newton-Krylov based fully-implicit solution procedure.

We have demonstrated method's capability to produce highly accurate solutions of difficult multiphysics problems on highly distorted unstructured grids. The scope of complexity involve multiple time scale problems, with wide spread of scales, including very fast (stiff) modes. In particular, we have shown all-speed flow capabilities, when fully-compressible simulations are performed for extremely small Mach numbers $< 10^{-3}$, without explicit filtering of acoustic modes on the problem formulation (governing equations) level. Also, we demonstrated an ability to incorporate stiff material strength models within the framework of seamless combination of fluid-solid multimaterial systems.

The future effort should be concentrated on two areas. First, on *improvement of preconditioning*. In the present study, we compute approximate Jacobian matrices, and utilize (incomplete) LU factorizations, as preconditioning techniques

for the GMRES based linear steps in Newton iterations. This is rather expensive in both memory requirements (for the Jacobian storage), and CPU time (for LU factorizations involved). Instead, we are looking for to develop and implement the Jacobian-free physics-based preconditioning strategy, in which operator-splitting legacy algorithms can be utilized to keep Krylov subspace limited. Second, we need to extend current modeling approach to work with *multi-material interfaces*. Our current plan in this direction is to involve a combination of the level set and volume tracking algorithms, to represent interfaces between (partially molten) powder material and ambient gas.

APPENDICES

This Page is Intentionally Left Blank

Appendix A

Tensor Calculus

SECOND rank tensors are often referred to simply as *tensors*. First rank tensors are referred to as *vectors*, and zero rank tensors as *scalars*.

The summation convection. If the same index letter appears twice in a term, summation with respect to that index is implied over the dimensions of the space. This rule *always* applies whenever an index is repeated in a term, for e.g.:

$$A_{kk} = A_{xx} + A_{yy} + A_{zz}$$
$$a_i A_{ji} = a_x A_{jx} + a_y A_{jy} + a_z A_{jz}$$

A.1 Vectors

Given two Cartesian vectors $\mathbf{a} = \left\{a_x, a_y, a_z\right\}^{\mathrm{T}}$ and $\mathbf{b} = \left\{b_x, b_y, b_z\right\}^{\mathrm{T}}$, the *dot product* is defined as

$$\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y + a_z b_z = a_k b_k \tag{A.1}$$

The *dyadic product* of two vectors is a tensor, denoted as [Ari]

$$\mathbf{ab} = \mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_x b_x & a_x b_y & a_x b_z \\ a_y b_x & a_y b_y & a_y b_z \\ a_z b_x & a_z b_y & a_z b_z \end{bmatrix} = a_k b_l \tag{A.2}$$

Spatial derivatives are denoted as

$$\nabla = \left\{\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right\}^{\mathrm{T}} = \left\{\partial_x, \partial_y, \partial_z\right\}^{\mathrm{T}} = \partial_j \tag{A.3}$$

Thus, the *gradient of* an arbitrary *scalar* φ is defined as

$$\nabla\varphi = \left\{ \frac{\partial\varphi}{\partial x}, \frac{\partial\varphi}{\partial y}, \frac{\partial\varphi}{\partial z} \right\}^{\mathrm{T}} = \left\{ \partial_x\varphi, \partial_y\varphi, \partial_z\varphi \right\}^{\mathrm{T}} = \partial_j\varphi \tag{A.4}$$

and, accordingly, dot product of a vector and a gradient of a scalar is

$$\mathbf{a} \cdot \nabla\varphi = a_x \frac{\partial\varphi}{\partial x} + a_y \frac{\partial\varphi}{\partial y} + a_z \frac{\partial\varphi}{\partial z} \tag{A.5}$$

Laplacian of an arbitrary *scalar* is defined as

$$\nabla \cdot \nabla\varphi = \nabla^2\varphi = \Delta\varphi = \frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} + \frac{\partial^2\varphi}{\partial z^2} = \partial_k^2\varphi \tag{A.6}$$

Divergence of a vector is defined as

$$\nabla \cdot \mathbf{a} = \frac{\partial a_x}{\partial x} + \frac{\partial a_y}{\partial y} + \frac{\partial a_z}{\partial z} \tag{A.7}$$

Gradient of an arbitrary *vector* is a tensor, defined as

$$\nabla\mathbf{a} = \begin{bmatrix} \frac{\partial a_x}{\partial x} & \frac{\partial a_x}{\partial y} & \frac{\partial a_x}{\partial z} \\ \frac{\partial a_y}{\partial x} & \frac{\partial a_y}{\partial y} & \frac{\partial a_y}{\partial z} \\ \frac{\partial a_z}{\partial x} & \frac{\partial a_z}{\partial y} & \frac{\partial a_z}{\partial z} \end{bmatrix} \tag{A.8}$$

and its *transpose*:

$$\nabla\mathbf{a}^{\mathrm{T}} = \begin{bmatrix} \frac{\partial a_x}{\partial x} & \frac{\partial a_y}{\partial x} & \frac{\partial a_y}{\partial x} \\ \frac{\partial a_x}{\partial y} & \frac{\partial a_y}{\partial y} & \frac{\partial a_y}{\partial y} \\ \frac{\partial a_x}{\partial z} & \frac{\partial a_y}{\partial z} & \frac{\partial a_y}{\partial z} \end{bmatrix} \tag{A.9}$$

Scalar product of a vector and divergence of a vector is a vector:

$$\mathbf{a} \cdot \nabla\mathbf{b} = \begin{Bmatrix} a_x \cdot \nabla b_x \\ a_y \cdot \nabla b_y \\ a_z \cdot \nabla b_z \end{Bmatrix} = \begin{Bmatrix} a_x \frac{\partial b_x}{\partial x} + a_y \frac{\partial b_x}{\partial y} + a_z \frac{\partial b_x}{\partial z} \\ a_x \frac{\partial b_y}{\partial x} + a_y \frac{\partial b_y}{\partial y} + a_z \frac{\partial b_y}{\partial z} \\ a_x \frac{\partial b_z}{\partial x} + a_y \frac{\partial b_z}{\partial y} + a_z \frac{\partial b_z}{\partial z} \end{Bmatrix} \tag{A.10}$$

Divergence of a dyadic product of two vectors is defined as

$$\nabla \cdot (\mathbf{ab}) = \nabla \cdot (\mathbf{a} \otimes \mathbf{b}) = \left\{ \begin{array}{l} \frac{\partial}{\partial x}\left(a_{x} b_{x}\right) + \frac{\partial}{\partial y}\left(a_{x} b_{y}\right) + \frac{\partial}{\partial z}\left(a_{x} b_{z}\right) \\ \frac{\partial}{\partial x}\left(a_{y} b_{x}\right) + \frac{\partial}{\partial y}\left(a_{y} b_{y}\right) + \frac{\partial}{\partial z}\left(a_{y} b_{z}\right) \\ \frac{\partial}{\partial x}\left(a_{z} b_{x}\right) + \frac{\partial}{\partial y}\left(a_{z} b_{y}\right) + \frac{\partial}{\partial z}\left(a_{z} b_{z}\right) \end{array} \right\} \quad \text{(A.11)}$$

A.2 Tensors

A component of a tensor is indicated by two indices, Thus, the i, j component of a tensor $\mathbb{A}$ is written as A_{ij}.

If $A_{ij} = A_{ji}$ for all i and j, the tensor is said to be *symmetric*. If $A_{ij} = -A_{ji}$ for all i and j, then it is *antisymmetric*. For an antisymmetric tensor, components in which i and j have the same value (diagonal) are zero.

The *transpose of a tensor* $\mathbb{A}$ is a tensor denoted as $\mathbb{A}^{T}$, with components

$$A^{T}_{ij} = A_{ji}$$

The *trace of a tensor* $\mathbb{A}$ is the scalar A_{kk}, e.g.

$$\text{trace}\,(\mathbb{A}) = A_{xx} + A_{yy} + A_{zz}$$

Addition/substraction of two tensors $\mathbb{A}$ and $\mathbb{B}$ is a tensor $\mathbb{C}$:

$$\mathbb{A} + \mathbb{B} = A_{ij} + B_{ij} = C_{ij} = \mathbb{C} \quad \text{(A.12)}$$

Direct product of a scalar c and tensor $\mathbb{A}$ is a tensor:

$$c\mathbb{A} = cA_{ij} = C_{ij} = \mathbb{C} \quad \text{(A.13)}$$

Inner product of two tensors $\mathbb{A}$ and $\mathbb{B}$ is a tensor $\mathbb{C}$:

$$\mathbb{A} \cdot \mathbb{B} = A_{ij} B_{kl} \delta_{jk} = A_{ij} B_{jl} = C_{il} = \mathbb{C} \quad \text{(A.14)}$$

Double inner product of two tensors $\mathbb{A}$ and $\mathbb{B}$ is a scalar:

$$\mathbb{A} : \mathbb{B} = A_{ij} B_{kl} \delta_{jk} \delta_{il} = A_{ij} B_{jl} \delta_{il} = A_{ij} B_{ji} \quad \text{(A.15)}$$

This Page is Intentionally Left Blank

Appendix B

Butcher Tableau for ESDIRK

In this appendix, we summarize coefficients for *"Explicit, Singly Diagonal Implicit Runge-Kutta" (ESDIRK)* time discretization scheme.

ESDIRK$_3$

$$
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
\frac{1767732205903}{2027836641118} & \frac{1767732205903}{4055673282236} & \frac{1767732205903}{4055673282236} & 0 & 0 \\
\frac{3}{5} & \frac{2746238789719}{10658868560708} & -\frac{640167445237}{6845629431997} & \frac{1767732205903}{4055673282236} & 0 \\
1 & \frac{1471266399579}{7840856788654} & -\frac{4482444167858}{7529755066697} & \frac{11266239266428}{11593286722821} & \frac{1767732205903}{4055673282236} \\
\hline
b_r & \frac{1471266399579}{7840856788654} & -\frac{4482444167858}{7529755066697} & \frac{11266239266428}{11593286722821} & \frac{1767732205903}{4055673282236} \\
\hline
\hat{b}_r & \frac{2756265671327}{12835298489170} & -\frac{10771552573575}{22201958757719} & \frac{9247589265047}{10645013368117} & \frac{2193209047091}{5459859503100}
\end{array}
\tag{B.1}
$$

ESDIRK$_4$

$$
\begin{array}{c|cccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\
\frac{83}{250} & \frac{8611}{62500} & -\frac{1743}{31250} & \frac{1}{4} & 0 & 0 & 0 \\
\frac{31}{50} & \frac{5012029}{34652500} & -\frac{654441}{2922500} & \frac{174375}{388108} & \frac{1}{4} & 0 & 0 \\
\frac{17}{20} & \frac{15267082809}{155376265600} & -\frac{71443401}{120774400} & \frac{730878875}{902184768} & \frac{2285395}{8070912} & \frac{1}{4} & 0 \\
1 & \frac{82889}{524892} & 0 & \frac{15625}{83664} & \frac{69875}{102672} & -\frac{2260}{8211} & \frac{1}{4} \\
\hline
\mathsf{b}_r & \frac{82889}{524892} & 0 & \frac{15625}{83664} & \frac{69875}{102672} & -\frac{2260}{8211} & \frac{1}{4} \\
\hline
\hat{\mathsf{b}}_r & \frac{4586570599}{29645900160} & 0 & \frac{178811875}{945068544} & \frac{814220225}{1159782912} & -\frac{3700637}{11593932} & \frac{61727}{225920}
\end{array}
\tag{B.2}
$$

ESDIRK$_5$

$$
\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
\frac{41}{100} & \frac{41}{200} & \frac{41}{200} & 0 & 0 \\
\frac{2935347310677}{11292855782101} & \frac{41}{400} & \frac{-567603406766}{11931857230679} & \frac{41}{200} & 0 \\
\frac{1426016391358}{7196633302097} & \frac{683785636431}{9252920307686} & 0 & \frac{-110385047103}{1367015193373} & \frac{41}{200} \\
\frac{92}{100} & \frac{3016520224154}{10081342136671} & 0 & \frac{30586259806659}{12414158314087} & \frac{-22760509404356}{11113319521817} \\
\frac{24}{100} & \frac{218866479029}{1489978393911} & 0 & \frac{638256894668}{5436446318841} & \frac{-1179710474555}{5321154724896} \\
\frac{3}{5} & \frac{1020004230633}{5715676835656} & 0 & \frac{25762820946817}{25263940353407} & -\frac{2161375909145}{9755907335909} \\
1 & \frac{-872700587467}{9133579230613} & 0 & 0 & \frac{22348218063261}{9555858737531} \\
\hline
\mathsf{b}_r & \frac{-872700587467}{9133579230613} & 0 & 0 & \frac{22348218063261}{9555858737531}
\end{array}
\qquad \mathbb{B} \qquad (B.3)
$$

$$
\mathbb{B} = \left[\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\frac{41}{200} & 0 & 0 & 0 \\
\frac{-60928119172}{8023461067671} & \frac{41}{200} & 0 & 0 \\
\frac{-211217309593}{5846859502534} & \frac{-4269925059573}{7827059040749} & \frac{41}{200} & 0 \\
\frac{-1143369518992}{8141816002931} & \frac{-39379526789629}{19018526304540} & \frac{32727382324388}{42900044865799} & \frac{41}{200} \\
\hline
\frac{-1143369518992}{8141816002931} & \frac{-39379526789629}{19018526304540} & \frac{32727382324388}{42900044865799} & \frac{41}{200}
\end{array}\right]
$$

This Page is Intentionally Left Blank

Appendix C

Thermodynamics

THe first law of thermodynamics [RP77, Atk94] can be stated as

$$dE = dQ + dW \tag{C.1}$$

where dE is the change in total energy of the system, dQ is the heat added to the system,

$$dQ = Td\mathcal{S}$$

and dW is the work done on the system. Hereafter, we denote thermodynamic entropy by $\mathcal{S}$. Simple substances are those for which the only important reversible work mode is volume change,

$$dW = -PdV \tag{C.2}$$

where V is the volume. With these, we can re-write eq.(C.1) as

$$d\mathfrak{u} = Td\mathfrak{s} + \frac{P}{\rho^2}d\rho \tag{C.3}$$

It is customary to express intensive thermodynamic variables in algebraic (or tabular) forms, known as *equations of state*,

$$\begin{gathered} \mathfrak{u} = \mathfrak{u}\left(\mathfrak{s}, \rho\right), \quad \text{or} \\ \mathfrak{h} = \mathfrak{h}\left(\mathfrak{s}, \rho\right), \quad \text{or} \\ P = P\left(\rho, \mathfrak{u}\right) \\ \text{etc.} \end{gathered} \tag{C.4}$$

Using the calculus of functions of two variables,

$$d\mathfrak{u} = \left.\frac{\partial \mathfrak{u}}{\partial \mathfrak{s}}\right|_{\rho} d\mathfrak{s} + \left.\frac{\partial \mathfrak{u}}{\partial \rho}\right|_{\mathfrak{s}} d\rho \tag{C.5}$$

From eq.(C.3) and (C.5),

$$T = \left.\frac{\partial \mathfrak{u}}{\partial \mathfrak{s}}\right|_{\rho} \tag{C.6}$$

and

$$P = \rho^2 \left.\frac{\partial \mathfrak{u}}{\partial \rho}\right|_{\mathfrak{s}} \tag{C.7}$$

where ps are coming from the equation of state (C.4).

Using the definition of enthalpy eq.(2.11), the first law of thermodynamics eq.(C.1) can be written as

$$d\mathfrak{h} = Td\mathfrak{s} + \upsilon dP \tag{C.8}$$

where $\upsilon = \frac{1}{\rho}$, which combined with

$$d\mathfrak{h} = \left.\frac{\partial \mathfrak{h}}{\partial \mathfrak{s}}\right|_{P} d\mathfrak{s} + \left.\frac{\partial \mathfrak{h}}{\partial P}\right|_{\mathfrak{s}} dP \tag{C.9}$$

gives another usefull set of relationships:

$$T = \left.\frac{\partial \mathfrak{h}}{\partial \mathfrak{s}}\right|_{P} \tag{C.10}$$

and

$$\rho = \left(\left.\frac{\partial \mathfrak{h}}{\partial P}\right|_{\mathfrak{s}}\right)^{-1} \tag{C.11}$$

For materials which do not allow phase transition (melting, freezing, evaporation, condensation), or in single-state (solid, liquid or vapour) zones of complex materials, pressure P and temperature T are always independent variables. This allows for some simplifications. In particular, using the calculus of functions of

two variables, for any three functions x, y and z, any two of which may be selected as the independent pair,

$$\left.\frac{\partial x}{\partial y}\right|_z \left.\frac{\partial y}{\partial z}\right|_x \left.\frac{\partial z}{\partial x}\right|_y = -1 \tag{C.12}$$

This allows to compute important ps of independent variables, especially useful for thermodynamical consistency of tabulated equations of state. For example,

- $\left.\frac{\partial P}{\partial T}\right|_\upsilon$ from $\upsilon = \upsilon\left(P, T\right)$, as

$$\left.\frac{\partial P}{\partial T}\right|_\upsilon \left.\frac{\partial T}{\partial \upsilon}\right|_P \left.\frac{\partial \upsilon}{\partial P}\right|_T = -1 \implies \left.\frac{\partial P}{\partial T}\right|_\upsilon = -\frac{\left.\frac{\partial \upsilon}{\partial T}\right|_P}{\left.\frac{\partial \upsilon}{\partial P}\right|_T} \tag{C.13}$$

- $\left.\frac{\partial P}{\partial \mathfrak{u}}\right|_\rho$ from $\rho = \rho\left(P, \mathfrak{u}\right)$:

$$\left.\frac{\partial P}{\partial \mathfrak{u}}\right|_\rho = -\frac{\left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_P}{\left.\frac{\partial \rho}{\partial P}\right|_\mathfrak{u}} \tag{C.14}$$

- $\left.\frac{\partial \upsilon}{\partial T}\right|_P$ from $P = P\left(\upsilon, T\right)$:

$$\left.\frac{\partial \upsilon}{\partial T}\right|_P = -\frac{\left.\frac{\partial P}{\partial T}\right|_\upsilon}{\left.\frac{\partial P}{\partial \upsilon}\right|_T} \tag{C.15}$$

C.1 Specific heats

Consider specific internal energy as a function of T and υ,

$$\mathfrak{u} = \mathfrak{u}\left(T, \upsilon\right)$$

The difference in energy between any two infinitesimally close states is then

$$d\mathfrak{u} = \left.\frac{\partial \mathfrak{u}}{\partial T}\right|_\upsilon dT + \left.\frac{\partial \mathfrak{u}}{\partial \upsilon}\right|_T d\upsilon \tag{C.16}$$

The slope of a line of constant υ on a $(\mathfrak{u} - T)$ thermodynamic plane is a function of state, and called *specific heat at constant volume*[1]:

$$C_\upsilon \equiv \left.\frac{\partial \mathfrak{u}}{\partial T}\right|_\upsilon \tag{C.17}$$

[1] Sometimes called *specific isochoric heat capacity*.

Using the calculus of functions of two variables[2],

$$C_v = \frac{1}{\left.\frac{\partial T}{\partial \mathfrak{u}}\right|_P - \frac{\left.\frac{\partial T}{\partial P}\right|_{\mathfrak{u}} \left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_P}{\left.\frac{\partial \rho}{\partial P}\right|_{\mathfrak{u}}}} \tag{C.21}$$

which can be used for tabulated equations of state.

Another specific heat is defined as

$$C_P \equiv \left.\frac{\partial \mathfrak{h}}{\partial T}\right|_P = \left.\frac{\partial \mathfrak{u}}{\partial T}\right|_P - \frac{P}{\rho^2} \left.\frac{\partial \rho}{\partial T}\right|_P \tag{C.22}$$

and called the *specific heat at constant pressure*[3]. Using the calculus of functions

[2] In fact,

$$\begin{aligned} d\rho &= \left.\tfrac{\partial \rho}{\partial P}\right|_{\mathfrak{u}} dP + \left.\tfrac{\partial \rho}{\partial \mathfrak{u}}\right|_P d\mathfrak{u} \\ dT &= \left.\tfrac{\partial T}{\partial P}\right|_{\mathfrak{u}} dP + \left.\tfrac{\partial T}{\partial \mathfrak{u}}\right|_P d\mathfrak{u} \\ dT &= \left.\tfrac{\partial T}{\partial \rho}\right|_{\mathfrak{u}} d\rho + \left.\tfrac{\partial T}{\partial \mathfrak{u}}\right|_\rho d\mathfrak{u} \end{aligned}$$

Eliminating dT in the last two equations, plugging $d\rho$ from the first one, and collecting the terms with dP on the left and with $d\mathfrak{u}$ on the right,

$$dP \left[\left.\tfrac{\partial T}{\partial P}\right|_{\mathfrak{u}} - \left.\tfrac{\partial T}{\partial \rho}\right|_{\mathfrak{u}} \left.\tfrac{\partial \rho}{\partial P}\right|_{\mathfrak{u}} \right] = d\mathfrak{u} \left[\left.\tfrac{\partial T}{\partial \mathfrak{u}}\right|_\rho - \left.\tfrac{\partial T}{\partial \mathfrak{u}}\right|_P + \left.\tfrac{\partial T}{\partial \rho}\right|_{\mathfrak{u}} \left.\tfrac{\partial \rho}{\partial \mathfrak{u}}\right|_P \right] \tag{C.18}$$

Since the changes $d\mathfrak{u}$ and dP are independent, setting $d\mathfrak{u} = 0$ leads to

$$\left.\frac{\partial T}{\partial \rho}\right|_{\mathfrak{u}} = \frac{\left.\frac{\partial T}{\partial P}\right|_{\mathfrak{u}}}{\left.\frac{\partial \rho}{\partial P}\right|_{\mathfrak{u}}} \tag{C.19}$$

Similarly, setting $dP = 0$ leads to

$$\left.\frac{\partial T}{\partial \mathfrak{u}}\right|_\rho = \left.\frac{\partial T}{\partial \mathfrak{u}}\right|_P - \left.\frac{\partial T}{\partial \rho}\right|_{\mathfrak{u}} \left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_P \tag{C.20}$$

which is inverse of equation (C.21).

[3] It is also sometimes called the *heat capacity at constant pressure*.

of two variables[4],

$$C_{_P} = \frac{1 - \frac{P}{\rho^2}\left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_{_P}}{\left.\frac{\partial T}{\partial \mathfrak{u}}\right|_{_P}} \tag{C.26}$$

Both $C_{_P}$ and $C_{_v}$ constitute two of the most important thermodynamic derivative functions, and their values have been experimentally determined as functions of the thermodynamic state for a tremendous number of simple compressible substances, [RP77, Atk94].

When specific heats are functions of both temperature and pressure, the material is defined as "imperfect gas". There are two important simplifications oftenly used in engineering applications:

- **Thermally perfect gas.** C_v is a function of temperature only.
- **Calorically perfect gas.** C_v is constant.

[4] In fact,

$$\begin{aligned} d\rho &= \left.\frac{\partial \rho}{\partial P}\right|_{\mathfrak{u}} dP + \left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_{P} d\mathfrak{u} \\ dT &= \left.\frac{\partial T}{\partial P}\right|_{\mathfrak{u}} dP + \left.\frac{\partial T}{\partial \mathfrak{u}}\right|_{P} d\mathfrak{u} \\ dT &= \left.\frac{\partial T}{\partial \rho}\right|_{\mathfrak{u}} d\rho + \left.\frac{\partial T}{\partial P}\right|_{\rho} dP \end{aligned}$$

Eliminating dT in the last two equations, plugging $d\rho$ from the first one, and collecting the terms with dP on the left and with $d\mathfrak{u}$ on the right,

$$dP\left[\left.\frac{\partial T}{\partial P}\right|_{\mathfrak{u}} - \left.\frac{\partial T}{\partial P}\right|_{\rho} - \left.\frac{\partial T}{\partial \rho}\right|_{P}\left.\frac{\partial \rho}{\partial P}\right|_{\mathfrak{u}}\right] = d\mathfrak{u}\left[\left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_{P}\left.\frac{\partial T}{\partial \rho}\right|_{P} - \left.\frac{\partial T}{\partial \mathfrak{u}}\right|_{P}\right] \tag{C.23}$$

Since the changes $d\mathfrak{u}$ and dP are independent, setting $dP = 0$ leads to

$$\left.\frac{\partial T}{\partial \rho}\right|_{P} = \frac{\left.\frac{\partial T}{\partial \mathfrak{u}}\right|_{P}}{\left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_{P}} \tag{C.24}$$

Similarly, setting $d\mathfrak{u} = 0$ leads to

$$\left.\frac{\partial T}{\partial P}\right|_{\rho} = \left.\frac{\partial T}{\partial P}\right|_{\mathfrak{u}} - \left.\frac{\partial T}{\partial \mathfrak{u}}\right|_{P} \frac{\left.\frac{\partial \rho}{\partial P}\right|_{\mathfrak{u}}}{\left(\left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_{P}\right.} \tag{C.25}$$

Combining eq.(C.24) and (C.22) leads to eq.(C.26).

C.2 Compressibilities

Consider specific volume as a function of T and P,

$$\upsilon = \upsilon\left(T, P\right)$$

The difference in specific volume between any two infinitesimally close states is then

$$d\upsilon = \left.\frac{\partial \upsilon}{\partial T}\right|_P dT + \left.\frac{\partial \upsilon}{\partial P}\right|_T dP$$

The slope $\left.\frac{\partial \upsilon}{\partial T}\right|_P$ represents the sensitivity of the specific volume to changes in temperature at constant pressure, leading to the definition of the *isobaric compressibility*:

$$\beta \equiv \frac{1}{\upsilon}\left.\frac{\partial \upsilon}{\partial T}\right|_P = -\frac{1}{\rho}\left.\frac{\partial \rho}{\partial T}\right|_P \tag{C.27}$$

Using eq.(C.33),

$$\beta = -\frac{1}{\rho}\frac{\left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_P}{\left.\frac{\partial T}{\partial \mathfrak{u}}\right|_P} \tag{C.28}$$

Using eq.(C.15),

$$\beta = -\frac{1}{\rho}\frac{\left.\frac{\partial P}{\partial T}\right|_\rho}{\left.\frac{\partial P}{\partial \rho}\right|_T} \tag{C.29}$$

The *coefficient of linear expansion* used in elementary strength-of-materials textbooks is

$$\alpha = \frac{1}{3}\beta$$

The slope $\left.\frac{\partial \upsilon}{\partial P}\right|_T$ is a measure of the change in specific volume associated with a change in pressure at constant temperature, defining the *isothermal compressibility*:

$$\chi \equiv -\frac{1}{\upsilon}\left.\frac{\partial \upsilon}{\partial P}\right|_T = \frac{1}{\rho}\left.\frac{\partial \rho}{\partial P}\right|_T \tag{C.30}$$

Young's modulus of elasticity is proportional to χ. Using the calculus of functions of two variables[5],

$$\chi = \frac{1}{\rho}\left[\left.\frac{\partial\rho}{\partial P}\right|_{\mathfrak{u}} - \frac{\left.\frac{\partial\rho}{\partial\mathfrak{u}}\right|_{P}\left.\frac{\partial T}{\partial P}\right|_{\mathfrak{u}}}{\left.\frac{\partial T}{\partial\mathfrak{u}}\right|_{P}}\right] \tag{C.34}$$

It is also possible to demonstrate that

$$\chi = \frac{C_{_P}}{\rho C_{_v} c^2} \tag{C.35}$$

C.3 Speed of sound

The sound speed of materials is defined as

$$c\left(\mathfrak{s}\right) = \sqrt{\left.\frac{\partial P}{\partial\rho}\right|_{\mathfrak{s}}} \tag{C.36}$$

[5] In fact,

$$\begin{aligned} d\rho &= \left.\tfrac{\partial\rho}{\partial P}\right|_{T} dP + \left.\tfrac{\partial\rho}{\partial T}\right|_{P} dT \\ d\rho &= \left.\tfrac{\partial\rho}{\partial P}\right|_{\mathfrak{u}} dP + \left.\tfrac{\partial\rho}{\partial\mathfrak{u}}\right|_{P} d\mathfrak{u} \\ dT &= \left.\tfrac{\partial T}{\partial P}\right|_{\mathfrak{u}} dP + \left.\tfrac{\partial T}{\partial\mathfrak{u}}\right|_{P} d\mathfrak{u} \end{aligned}$$

Eliminating $d\rho$ in the first two equations, plugging into that the last equation for dT, and collecting terms for dP on the left and for $d\mathfrak{u}$ on the right,

$$dP\left[\left.\tfrac{\partial\rho}{\partial P}\right|_{T} - \left.\tfrac{\partial\rho}{\partial P}\right|_{\mathfrak{u}} + \left.\tfrac{\partial\rho}{\partial T}\right|_{P}\left.\tfrac{\partial T}{\partial P}\right|_{\mathfrak{u}}\right] = d\mathfrak{u}\left[\left.\tfrac{\partial\rho}{\partial\mathfrak{u}}\right|_{P} - \left.\tfrac{\partial\rho}{\partial T}\right|_{P}\left.\tfrac{\partial T}{\partial\mathfrak{u}}\right|_{P}\right] \tag{C.31}$$

Since the changes $d\mathfrak{u}$ and dP are independent, setting $d\mathfrak{u} = 0$ leads to

$$\left.\frac{\partial\rho}{\partial P}\right|_{T} = \left.\frac{\partial\rho}{\partial P}\right|_{\mathfrak{u}} - \left.\frac{\partial\rho}{\partial\mathfrak{u}}\right|_{P}\frac{\left.\frac{\partial T}{\partial P}\right|_{\mathfrak{u}}}{\left.\frac{\partial T}{\partial\mathfrak{u}}\right|_{P}} \tag{C.32}$$

Similarly, setting $dP = 0$ generates

$$\left.\frac{\partial\rho}{\partial T}\right|_{P} = \frac{\left.\frac{\partial\rho}{\partial\mathfrak{u}}\right|_{P}}{\left.\frac{\partial T}{\partial\mathfrak{u}}\right|_{P}} \tag{C.33}$$

Combining eq.(C.32) with eq.(C.30) leads to eq.(C.34).

It is also related to partial derivatives of the pressure [Atk94], $\left.\frac{\partial P}{\partial \rho}\right|_{\mathfrak{u}}$ and $\left.\frac{\partial P}{\partial \mathfrak{u}}\right|_{\rho}$:

$$c\left(\rho,\mathfrak{u}\right)=\sqrt{\left.\frac{\partial P}{\partial \rho}\right|_{\mathfrak{u}}+\frac{P\left.\frac{\partial P}{\partial \mathfrak{u}}\right|_{\rho}}{\rho^{2}}} \tag{C.37}$$

or

$$c\left(\rho,T\right)=\sqrt{\left.\frac{\partial P}{\partial \rho}\right|_{T}+\frac{T\left(\left.\frac{\partial P}{\partial T}\right|_{\rho}\right)^{2}}{C_{v}\rho^{2}}} \tag{C.38}$$

Using eq.(C.14),

$$c\left(P,\mathfrak{u}\right)=\sqrt{\frac{1}{\left.\frac{\partial \rho}{\partial P}\right|_{\mathfrak{u}}}\left(1-\frac{P\left.\frac{\partial \rho}{\partial \mathfrak{u}}\right|_{P}}{\rho^{2}}\right)} \tag{C.39}$$

C.4 Entropy

Specific *entropy* change $d\mathfrak{s}$ is defined by

$$d\mathfrak{s}=\frac{d\mathfrak{u}}{T}+\frac{P}{T}dv \tag{C.40}$$

and

$$d\mathfrak{s}=\frac{d\mathfrak{h}}{T}-\frac{v}{T}dP \tag{C.41}$$

Treating T and v as independent variables,

$$d\mathfrak{s}=\left.\frac{\partial \mathfrak{s}}{\partial T}\right|_{v}dT+\left.\frac{\partial \mathfrak{s}}{\partial v}\right|_{T}dv \tag{C.42}$$

Next, using Maxwell relations (see [MHDM11]):

$$\begin{array}{rcl}\left.\frac{\partial T}{\partial v}\right|_{\mathfrak{s}} & = & -\left.\frac{\partial P}{\partial \mathfrak{s}}\right|_{v}\\ \left.\frac{\partial T}{\partial P}\right|_{\mathfrak{s}} & = & \left.\frac{\partial v}{\partial \mathfrak{s}}\right|_{P}\\ \left.\frac{\partial P}{\partial T}\right|_{v} & = & \left.\frac{\partial \mathfrak{s}}{\partial v}\right|_{T}\\ \left.\frac{\partial v}{\partial T}\right|_{P} & = & -\left.\frac{\partial \mathfrak{s}}{\partial P}\right|_{T}\end{array} \tag{C.43}$$

one can write:

$$d\mathfrak{s} = \left.\frac{\partial \mathfrak{s}}{\partial T}\right|_{\upsilon} dT + \left.\frac{\partial P}{\partial T}\right|_{\upsilon} d\upsilon \tag{C.44}$$

It can be shown that

$$\left.\frac{\partial \mathfrak{s}}{\partial T}\right|_{\upsilon} = \frac{C_{\upsilon}}{T} \tag{C.45}$$

(see p.652 of [MHDM11]). Thus,

$$d\mathfrak{s} = \frac{C_{\upsilon}}{T} dT + \left.\frac{\partial P}{\partial T}\right|_{\upsilon} d\upsilon \tag{C.46}$$

C.5 γ-law gas

Equation of state for an *ideal* or *perfect gas* is

$$P = \rho R T \tag{C.47}$$

where $R = \frac{R_u}{M}$ is the specific gas constant, with the universal gas constant $R_u \approx 8.31451 \frac{\mathrm{J}}{\mathrm{mol \cdot K}}$ and M is the molecular weight of the gas.

One of the important features of a perfect gas is that its internal energy depends only upon its temperature. Thus, eq.(C.16) reduces to:

$$d\mathfrak{u} = C_{\upsilon}(T) dT \tag{C.48}$$

and C_{υ} depends only upon T. From eq.(2.11), the enthalpy $\mathfrak{h}$ also depends only on temperature, leading to

$$d\mathfrak{h} = C_{P}(T) dT \tag{C.49}$$

Next, one can easily see that

$$d\mathfrak{h} = C_{P}(T) dT = d\mathfrak{u} + d\left(P\upsilon\right) = C_{\upsilon} dT + R dT$$

leading to

$$R = C_p - C_{\upsilon}$$

Now, we can introduce γ as the ratio of specific heats [Atk94]:

$$\gamma \equiv \frac{C_{_P}}{C_{_v}} \tag{C.50}$$

The next level of assumption is stating that $C_{_v}$ is independent of temperature, i.e. the assumption of *calorically perfect gas*, leading to the following relation of the specific internal energy and temperature:

$$\mathfrak{u}(T) = \mathfrak{u}_0 + C_{_v}\left(T - T_0\right) \tag{C.51}$$

where $\mathfrak{u}_0$ and T_0 are some reference constants.

With this, we can write ***"γ-law"*** equation of state as

$$P\left(\rho, \mathfrak{u}\right) = \rho(\gamma - 1)\left(\mathfrak{u} - \mathfrak{u}_0 + C_{_v}T_{_0}\right) \tag{C.52}$$

and important thermodynamic properties are defined as

$C_{_v}$	$=$	$const$	Specific heat at constant volume
$C_{_P}$	$=$	$\gamma C_{_v}$	Specific heat at constant pressure
$\beta(T)$	$=$	$\frac{1}{T}$	Isobaric compressibility
$\chi(P)$	$=$	$\frac{1}{P}$	Isothermal compressibility
$c\left(P,\rho\right)$	$=$	$\sqrt{\frac{\gamma P}{\rho}}$	Speed of sound
$\left.\frac{\partial P}{\partial \mathfrak{u}}\right\|_{\rho}$	$=$	$\rho(\gamma - 1)$	
$\left.\frac{\partial P}{\partial \rho}\right\|_{\mathfrak{u}}$	$=$	$\frac{P}{\rho}$	

(C.53)

Entropy is defined integrating eqs.(C.40) or (C.41), as follows. Since

$$d\mathfrak{u} = C_{_v}dT$$

$$d\mathfrak{h} = C_{_p}dT = \gamma C_{_v}dT$$

and

$$P\upsilon = C_{_v}\left(\gamma - 1\right)T$$

then

$$d\mathfrak{s} = \frac{C_v dT}{T} + C_v \left(\gamma - 1\right) \frac{d\upsilon}{\upsilon}$$

and

$$d\mathfrak{s} = \frac{C_v \gamma dT}{T} - C_v \left(\gamma - 1\right) \frac{dP}{P}$$

Introducing "reference" entropy $\mathfrak{s}_0 = \mathfrak{s}\left(T_0, \rho_0\right) = \mathfrak{s}\left(T_0, P_0\right)$, specific entropy of γ-gas can be defined as:

$$\begin{aligned} \mathfrak{s}\left(T, v\right) &= \mathfrak{s}_0 + \int_{T_0}^{T} \frac{C_v}{T} dT + \int_{\upsilon_0}^{\upsilon} \frac{C_v(\gamma-1)}{\upsilon} d\upsilon \\ &= \mathfrak{s}_0 + C_v \left(\ln \frac{T}{T_0} + (\gamma - 1) \ln \frac{\upsilon}{\upsilon_0}\right) \end{aligned} \tag{C.54}$$

and, similarly:

$$\mathfrak{s}\left(T, P\right) = \mathfrak{s}_0 + C_v \left(\gamma \ln \frac{T}{T_0} - (\gamma - 1) \ln \frac{P}{P_0}\right) \tag{C.55}$$

C.6 2-parameter fluid

The following equation of state is used for nearly-incompressible flows:

$$P\left(\rho\right) = P_0 + A \frac{\rho - \rho_0}{\rho_0} \tag{C.56}$$

where P_0 and ρ_0 are reference pressure and density, respectively. The speed of sound is defined by const A, as

$$c = \sqrt{\frac{A}{\rho_0}} \tag{C.57}$$

The fluid is assumed to be calorically perfect, eq.(C.51). Since pressure and density are not functions of energy,

$$C_v = \left.\frac{\partial \mathfrak{u}}{\partial T}\right|_{\rho} = \left.\frac{\partial \mathfrak{h}}{\partial T}\right|_{P} = C_P \tag{C.58}$$

and $\gamma = 1$.

This Page is Intentionally Left Blank

Bibliography

[ABC00] Ann S. Almgren, John B. Bell, and William Y. Crutchfield. Approximate projection methods: Part i. inviscid analysis. *SIAM Journal on Scientific Computing*, 22(4):1139–1159, 2000.

[ABCH93] Ann S. Almgren, John B. Bell, Phillip Colella, and Louis H Howell. An adaptive projection method for the incompressible euler equations. In *Eleventh AIAA Computational Fluid Dynamics Conference*, pages 530–539. AIAA, 1993.

[ABS96] Ann S. Almgren, John B. Bell, and William G. Szymcyzk. A numerical method for the incompressible navier-stokes equations based on an approximate projection. *SIAM Journal for Scientific Computing*, 17(2):358–369, March 1996.

[ale13] ALE3D Web page, 2013. https://wci.llnl.gov/simulation/computer-codes/ale3d.

[ALE14] Team ALE3D. ALE3D users manual, An Arbitrary Lagrangian/Eulerian 2D and 3D Code System. Technical Report LLNL-SM-650174 - Version 4.22.x, Lawrence Livermore National Laboratory, January 31 2014.

[Ari] R. Aris. *Vectors, Tensors, and the Basic Equations of Fluid Mechanics*. Dover Publications, Inc., New York.

[Atk94] P. Atkins. *Physical Chemistry*. Freeman, New York, 5th edition, 1994.

[BBE+04] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical

Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[BBG+01] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2001. http://www.mcs.anl.gov/petsc.

[BCG89] John B. Bell, Philip Colella, and Harland M. Glaz. A second-order projection method for the incompressible navier-stokes equations. *Journal of Computational Physics*, 85:257–283, 1989.

[BCVK02] H. Bijl, M.H. Carpenter, V.N. Vatsa, and C.A. Kennedy. Implicit time integration schemes for the unsteady compressible Navier-Stokes equations: Laminar flow. *Journal of Computational Physics*, 179:313–329, 2002.

[BGMS97] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[BJ89] T.J. Barth and P.O. Jesperson. The design and application of upwind schemes on unstructured meshes. In *AISS-90-0013, AIAA 28th Aerospace Sciences Meeting*, Reno, Nevada, USA, January 1989.

[BM95] David L. Brown and Michael L. Minion. Performance of underresolved two-dimensional incompressible flow simulations. *Journal of Computational Physics*, 122:165–183, 1995.

[BS90] P.N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM Journal of Science and Statistical Computing*, 11:450–480, 1990.

[Cho67] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2:12–26, 1967.

[Cho68] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computations*, 22:745–762, 1968.

[CHS90] B. Cockburn, S. Hou, and C. W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: The multidimensional case. *Mathematics of Computation*, 54:545–581, 1990.

[CKB+05] M.H. Carpenter, C.A. Kennedy, H. Bijl, S.A. Vilken, and V.N. Vatsa. Fourth-order Runge-Kutta schemes for fluid mechanics applications. *SIAM Journal of Scientific Computing*, 25:157–194, 2005.

[CLS89] B. Cockburn, S.-Y. Lin, and C. W. Shu. TVB Runge-Kutta local projection Discontinuous Galerkin finite element method for conservation laws III: One-dimensional systems. *Journal of Computational Physics*, 84:90–113, 1989.

[Coc89] B. Cockburn. *Introduction to Discontinuous Galerkin method for Convection-Dominated Problems*, volume 1697 of *Lecture Notes in Mathematics*. Springer, Berlin/Heidelberg, 1989.

[CS89] B. Cockburn and C. W. Shu. TVB Runge-Kutta local projection Discontinuous Galerkin finite element method for conservation laws II: General framework. *Mathematics of Computation*, 52:411–435, 1989.

[DBNS96] T. Dinh, V.A. Bui, R. Nourgaliev, and B.R. Sehgal. Crust dynamics under PWR in-vessel melt retention conditions. In *ANS Proceedings of the 1996 National Heat Transfer Conference*, volume HTC-Vol.9, pages 368–375, Houston, Texas, USA, August 1996.

[DBTM08] M. Dumbster, D.S. Balsara, E.F. Toro, and C.D. Munz. A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes. *Journal of Computational Physics*, 227:8209–8253, 2008.

[de 83] G. de Vahl Davis. Natural convection of air in a square cavity: a bench mark numerical solution. *International Journal for Numerical Methods in Fluids*, 3:249–264, 1983.

[dJ83] G. de Vahl Davis and I. P. Jones. Natural convection in a square cavity: a comparison exercise. *International Journal for Numerical Methods in Fluids*, 3:227–248, 1983.

[DS83] Jr. Dennis, J.E and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.

[Dum10] M. Dumbster. Arbitrary high order $p_n p_m$ schemes on unstructured meshes for the compressible Navier-Stokes equations. *Computers & Fluids*, 39:60–76, 2010.

[DZ09] M. Dumbster and O. Zanotti. Very high order $p_n p_m$ schemes on unstructured meshes for the resistive relativistic MHD equations. *Journal of Computational Physics*, 228:6991–7006, 2009.

[EW96] S.C. Eisenstat and H.F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM Journal of Science and Statistical Computing*, 17:16–32, 1996.

[Fle91] C.A.J. Fletcher. *Computational Techniques for Fluid Dynamics. Fundamental and General Techniques*, volume 1. Springer-Verlag, Berlin, Heidelberg, New-York, 1991.

[GC90] Philip M. Gresho and Stevens T. Chan. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 2: Implementation. *International Journal for Numerical Methods in Fluids*, 11:621–659, 1990.

[GC96] Philip M. Gresho and Stevens T. Chan. Projection 2 goes turbulent – and fully implicit. *preprint International Journal for Computational Fluid Dynamics*, March 1996. (LLNL UCRL-JC-123727).

[GCCH95] Philip M. Gresho, Stevens T. Chan, Mark A. Christon, and Allen C. Hindmarsh. A little more on stabilized $q1q1$ for transient viscous incompressible flow. *International Journal for Numerical Methods in Fluids*, 21:837–856, 1995.

[GGS82] U. Ghia, K.N. Ghia, and C.T Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a Multigrid method. *Journal of Computational Physics*, 48:347–411, 1982.

[GQ97] J.-L. Guermond and L. Quartapelle. Calculation of incompressible viscous flow by an unconditionally stable projection fem. *Journal of Computational Physics*, 132:12–23, 1997.

[GQ98a] Jean-Luc Guermond and L. Quartapelle. On stability and convergence of projection methods based on pressure poisson equation. *International Journal for Numerical Methods in Fluids*, 26:1039–1053, 1998.

[GQ98b] Jean-Luc Guermond and L. Quartapelle. On the approximation of the unsteady navier-stokes equations by finite element projection methods. *Numerische Mathematik*, 80:207–238, 1998.

[Gre90] Philip M. Gresho. On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 1: Theory. *International Journal for Numerical Methods in Fluids*, 11:587–620, 1990.

[Gue96] Jean-Luc Guermond. Some implementations of projection methods for navier-stokes equations. *Mathematical Modelling and Numerical Analysis*, 30(5):637–667, 1996.

[Gue97] Jean-Luc Guermond. A convergence result for the approximation of the navier-stokes equations by an incremental projection method. *C. R. Acad. Sci. Paris*, 325:1329–1332, 1997.

[HA68] F. H. Harlow and A. A. Amsden. Numerical calculation of almost incompressible flow. *Journal of Computational Physics*, 3:80–93, 1968.

[HA71] F. H. Harlow and A. A. Amsden. A numerical fluid dynamics calculation method for all flow speeds. *Journal of Computational Physics*, 8:197–213, 1971.

[HA75a] F. H. Harlow and A. A. Amsden. Flow of interpenetrating material phases. *Journal of Computational Physics*, 18:440–464, 1975.

[HA75b] F. H. Harlow and A. A. Amsden. Numerical calculation of multiphase fluid flow. *Journal of Computational Physics*, 17:19–52, 1975.

[Iss85] R. I. Issa. Solution of the Implicitely Discretized Fluid Flow Equations by Operator-Splitting. *Journal of Computational Physics*, 62:40–65, 1985.

[Kan86] J. Van Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM Journal for Scientific and Statistical Computing*, 7:870–891, 1986.

[Ker81] D.S. Kershaw. Differencing of the diffusion equation in lagrangian hydrodynamic codes. *Journal of Computational Physics*, 39:375–395, 1981.

[KK04] D. A. Knoll and D. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *Journal of Computational Physics*, 193:357–397, 2004.

[KNW99] Omar M. Knio, Habib N. Najm, and Peter S. Wyckoff. A semi-implicit numerical scheme for reacting flow. ii. stiff operator-split formulation. *pre-print submitted to Journal of Computational Physics*, 1999.

[LBL06] H. Luo, J.D. Baum, and R. Löhner. A fast, p-Multigrid Discontinuous Galerkin method for the Euler equations on unstructured grids. *Journal of Computational Physics*, 211(2):767–783, 2006.

[LBL08] H. Luo, J.D. Baum, and R. Löhner. A Discontinuous Galerkin method based on a Taylor basis for the compressible flows on arbitrary grids. *Journal of Computational Physics*, 227(20):8875–8893, 2008.

[Li05] B.Q. Li. *Discontinuous Finite Elements in Fluid Dynamics and Heat Transfer*. Springer, 2005.

[LL88] L.D. Landau and E.M. Lifschitz. *Hydrodynamics, Theoretical Physics*, volume VI. Nauka, Moscow, 4th edition, 1988.

[LLN12] H. Luo, L. Luo, and R.R. Nourgaliev. A Reconstructed Discontinuous Galerkin method for the Euler equations on arbitrary grids. *Communication in Computational Physics*, 12(5):1495–1519, November 2012.

[LLNC11] H. Luo, L. Luo, R.R. Nourgaliev, and C. Cai. A parallel, Reconstructed Discontinuous Galerkin method for the compressible flows on arbitrary grids. *Communication in Computational Physics*, 9(2):363–389, February 2011.

[LLNM09] H. Luo, L. Luo, R.R. Nourgaliev, and V. Mousseau. A Reconstructed Discontinuous Galerkin method for the compressible Euler equations on arbitrary grids. In *19th AIAA Computational Fluid Dynamics Conference*, San Antonio, Texas, USA, June 22-25 2009. AIAA 2009-3788.

[LLNM10a] H. Luo, L. Luo, R. Norgaliev, and V. A. Mousseau. A parallel Reconstructed Discontinuous Galerkin method for compressible flows on arbitrary grids. 2010. AIAA-2010-0366.

[LLNM10b] H. Luo, L. Luo, R. Norgaliev, and V. A. Mousseau. A Reconstructed Discontinuous Galerkin method for the compressible Navier-Stokes equations on arbitrary grids. 2010. AIAA-2010-0364.

[LLNM10c] H. Luo, L. Luo, R.R. Nourgaliev, and V. Mousseau. A Reconstructed Discontinuous Galerkin method for the compressible Navier-Stokes equations on arbitrary grids. *Journal of Computational Physics*, 229:6961–6978, 2010.

[LvL09] M. Lo and B. van Leer. Analysis and implementation of Recovery-based Discontinuous Galerkin for diffusion. In *19th AIAA Computational Fluid Dynamics Conference*, San Antonio, Texas, USA, June 22-25 2009. AIAA 2009-3786.

[LXL+12] H. Luo, Y. Xia, S. Li, R.R. Nourgaliev, and C. Cai. A Hermite WENO Reconstruction-based Discontinuous Galerkin method for the Euler equations on tetrahedral grids. *Journal of Computational Physics*, 231:5489–5502, 2012.

[LXS+13] H. Luo, Y. Xia, S. Spiegel, R.R. Nourgaliev, and Z. Jiang. A Reconstructed Discontinuous Galerkin method based on a Hierarchical WENO reconstruction for compressible flows on tetrahedral grids. *Journal of Computational Physics*, 236:477–492, 2013.

[MB97] Michael L. Minion and David L. Brown. Performance of under-resoloved two-dimensional incompressible flow simulations, ii. *Journal of Computational Physics*, 138:734–765, 1997.

[MHDM11] Moran M.J., Shapiro H.N., Boettner D.D., and Bailey M.B. *Fundamentals of Engineering Thermodynamics*. John Wiley & Sons, Inc, 2011.

[Min96] Michael L. Minion. A projection method for locally refined grids. *Journal of Computational Physics*, 127:158–178, 1996.

[Nou98] R.R. Nourgaliev. Modeling and analysis of heat and mass transfer processes during in-vessel melt progression stage of Light Water Reactor (LWR) severe accidents. Technical Report ISBN 91-7170-235-0, Royal Institute of Technology, Department of Nuclear Power Safety, Stockholm, Sweden, April 27 1998. Doctoral Thesis.

[NPM09] R.R. Nourgaliev, H.-K. Park, and V. A. Mousseau. Recovery Discontinuous Galerkin Jacobian-free Newton-Krylov method for multiphysics problems. *Computational Fluid Dynamics Review 2008*, 2009. In press.

[NTP+08] R.R. Nourgaliev, T.G. Theofanous, H. Park, V. Mousseau, and D. Knoll. Direct Numerical Simulation of interfacial flows: Sharp-Interface Method (I-SIM). In *46th AIAA Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, USA, January 2008. AIAA-2008-1453.

[OI01] P. J. Oliveira and R. I. Issa. An improved PISO algorithm for the computation of buoyancy-driven flows. *Numerical Heat Transfer, Part B*, 40:473–493, 2001.

[PAB+97] Elbridge G. Puckett, Ann S. Almgren, John B. Bell, Daniel L. Marcus, and William J. Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics*, 130:269–282, 1997.

[Pat80] S. Patankar. *Numerical Heat Transfer and Fluid Flow*. Taylor & Francis, 1980.

[PS72] S.V. Patankar and D.B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat Mass Transfer*, 15:1787–1806, 1972.

[PW98] M. Pernice and H.F. Walker. NITSOL: A Newton iterative solver for nonlinear systems. *SIAM Journal of Science and Statistical Computing*, 19:302–318, 1998.

[QS03] J. Qui and C.-W. Shu. Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method: one-dimensional case. *Journal of Computational Physics*, 193:115–135, 2003.

[Reb07] B. Rebourcet. Some remarks on Kershaw's legacy diffusion scheme. In *Workshop on Numerical methods for multi-material fluid flows*, Prague, Czech Republic, September 2007.

[RH73] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory Report, 1973.

[Rid94a] William J. Rider. Filtering nonsolenoidal modes in numerical solutions of incompressible flows. Technical Report LA-UR-3014, Los Alamos National Laboratory, Los Alamos, New Mexico, September 1994.

[Rid94b] William J. Rider. The robust formulation of approximate projection methods for incompressible flows. Technical Report LA-UR-3015, Los Alamos National Laboratory, 1994.

[Rid95] William J. Rider. Approximate projection methods for incompressible flow: implementation, variants and robustness. Technical Report LA-UR-2000, Los Alamos National Laboratory, Los Alamos, New Mexico, July 1995.

[RKM+95] W. J. Rider, D. B. Kothe, S. J. Mosso, J. H. Cerutti, and J. I. Hochstein. Accurate solution algorithms for incompressible multiphase flows. Technical Report AIAA-95-0699, AIAA, Reno, Nevada, January 1995.

[RP77] W. C. Reynolds and H. C. Perkins. *Engineering Thermodynamics*. McGraw-Hill, Inc., 2nd edition, 1977.

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2nd edition, 2003.

[SAB+99] M. Sussman, A.S. Almgren, J.B. Bell, P. Colella, and L.H. Howell. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, 148:81–124, 1999.

[SBG96] B. Smith, P. Bjørstad, and W. Gropp. *Domain Decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[SS86] Y. Saad and M.H. Schultz. GMRES: a Generalized Minimal Residual algorithm for solving linear systems. *SIAM Journal of Science and Statistical Computing*, 7:856, 1986.

[Tor99] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics. A practical Introduction*. Springer, Berlin/Heidelberg, 2nd edition, 1999.

[vLL09] B. van Leer and M. Lo. Unification of Discontinuous Galerkin methods for Advection and Diffusion. In *47th AIAA Aerospace Sciences Meeting and Exhibit*, Orlando, Florida, USA, 2009. AIAA 2009-0400.

[vLLR07] B. van Leer, M. Lo, and M. Raalte. Discontinuous Galerkin for diffusion based on recovery. In *18th AIAA Computational Fluid Dynamics Conference*, Miami, Florida, USA, June 26-28 2007. AIAA 2007-4083.

[vLN05] B. van Leer and S. Nomura. Discontinuous Galerkin for diffusion. In *17th AIAA Computational Fluid Dynamics Conference*, Toronto, Ontario, Canada, June 6-9 2005. AIAA 2005-5108.

[VP87] V.R. Voller and C. Prakash. A fixed grid numerical modelling methodology for convection-diffusion mushy region phase-change problems. *International Journal of Heat Mass Transfer*, 30(8):1709–1719, 1987.

[vRvL08] M. va Raalte and B. van Leer. Bilinear forms for the recovery-based Discontinuous Galerkin method for diffusion. *Communications in Computational Physics*, 5:683–693, 2008.

[Wet98] Brian B. Wetton. Error analysis of pressure increment schemes. *submitted to SINUM*, April 1998.

[XLFN14] Y. Xia, H. Luo, M. Frisbey, and R.R. Nourgaliev. A set of parallel, implicit methods for Reconstructed Discontinuous Galerkin method for compressible flows on 3d hybrid grids. *Computers & Fluids*, 96:406–421, 2014.

[XLN14] Y. Xia, H. Luo, and R.R. Nourgaliev. An Implicit Hermite WENO Reconstruction-based Discontinuous Galerkin on tetrahedral grids. *Computers & Fluids*, 98:134–151, 2014.

[YNK+11] R. Youngblood, R. Nourgaliev, D. Kelly, C. Smith, and T.-N. Dinh. Framework for applying a Next-Generation Safety Analysis Code to plant life extension decision-making. In *ANS Proceedings of the 2011 ANS Summer Meeting*, June 2011.

This Page is Intentionally Left Blank

Index

This Page is Intentionally Left Blank